NAVAL POSTGRADUATE SCHOOL

Monterey, California

THESIS

ADAPTIVE ALGORITHMS FOR TWO DIMENSIONAL FILTERING

by

Steven L. Wilstrup

September 1988

Thesis Advisor: Murali Tummala

Approved for public release; distribution is unlimited

Unclassified

security classification of this page

REPORT DOCUMENTATION PAGE

1a Report Security Classification: Unclassified		1b Restrictive Markings	
2a Security Classification Authority		3 Distribution/Availability of Report: Approved for public release; distribution is unlimited.	
2b Declassification Downgrading Schedule			
4 Performing Organization Report Number(s)		5 Monitoring Organization Report Number(s)	
6a Name of Performing Organization: Naval Postgraduate School	6b Office Symbol (*if applicable*): 32	7a Name of Monitoring Organization: Naval Postgraduate School	
6c Address (*city, state, and ZIP code*): Monterey, CA 93943-5000		7b Address (*city, state, and ZIP code*): Monterey, CA 93943-5000	
8a Name of Funding/Sponsoring Organization	8b Office Symbol (*if applicable*)	9 Procurement Instrument Identification Number	
8c Address (*city, state, and ZIP code*)		10 Source of Funding Numbers	
		Program Element No	Project No / Task No / Work Unit Accession No

11 Title (*Include security classification*): ADAPTIVE ALGORITHMS FOR TWO DIMENSIONAL FILTERING

12 Personal Author(s): Steven L. Wilstrup

13a Type of Report	13b Time Covered	14 Date of Report (*year, month, day*)	15 Page Count
Master's Thesis	From To	September 1988	75

16 Supplementary Notation: The views expressed in this thesis are those of the author and do not reflect the official policy or position of the Department of Defense or the U.S. Government.

17 Cosati Codes			18 Subject Terms (*continue on reverse if necessary and identify by block number*)
Field	Group	Subgroup	Adaptive, Filter, Wiener, LMS, RLS, Noise canceler, ALE

19 Abstract (*continue on reverse if necessary and identify by block number*)

In this thesis, an adaptive two dimensional least mean squares (LMS) algorithm and a recursive least squares (RLS) algorithm are developed from the one dimensional algorithms. Design of the two dimensional LMS and RLS algorithms are studied for accuracy based on the results of a two dimensional system identification model which was used in testing the algorithms. Application of the two algorithms is demonstrated through computer simulation in which the adaptive filters are employed in a noise canceler and an adaptive line enhancer and applied to an image processing problem.

... Wiener filters, one dimensional ... (ICR)

20 Distribution/Availability of Abstract	21 Abstract Security Classification	
☒ unclassified unlimited ☐ same as report ☐ DTIC users	Unclassified	
22a Name of Responsible Individual: Murali Tummala	22b Telephone (*Include Area code*): (408) 646-2645	22c Office Symbol: 62Tu

DD FORM 1473,84 MAR

83 APR edition may be used until exhausted
All other editions are obsolete

security classification of this page

Unclassified

Approved for public release; distribution is unlimited.

Adaptive Algorithms for Two
Dimensional Filtering

by

Steven L. Wilstrup
Lieutenant Commander, United States Navy
M.B.A., University of West Florida, 1983

Submitted in partial fulfillment of the
requirements for the degree of

MASTER OF SCIENCE IN ELECTRICAL ENGINEERING

from the

NAVAL POSTGRADUATE SCHOOL
September 1988

Author: Steven L. Wilstrup
Steven L. Wilstrup

Approved by: Murali Tummala
Murali Tummala, Thesis Advisor

Charles W. Therrien
Charles W. Therrien, Second Reader

John C Powers
John P. Powers, Chairman,
Department of Electrical and Computer Engineering

G E Schacher
Gordon E. Schacher,
Dean of Science and Engineering

ABSTRACT

In this thesis, an adaptive two dimensional least mean squares (LMS) algorithm and a recursive least squares (RLS) algorithm are developed from the one dimensional algorithms. Design of the two dimensional LMS and RLS algorithms are studied for accuracy based on the results of a two dimensional system identification model which was used in testing the algorithms. Application of the two algorithms is demonstrated through computer simulation in which the adaptive filters are employed in a noise canceler and an adaptive line enhancer and applied to an image processing problem.

TABLE OF CONTENTS

LIST OF TABLES

LIST OF FIGURES

I. INTRODUCTION

The area of digital signal processing has experienced a rapid growth in the last decade. Reasons for this have been the tremendous advances in integrated circuit technology, and some significant developments in digital processing techniques achieved during this period. Included in these developments are methods which extend certain one dimensional digital signal processing techniques to two dimensions. This extension is by no means trivial. Three significant factors must be considered:

1. more degrees of freedom are available in a two dimensional system; this gives a system designer more flexibility than the one dimensional case,
2. one dimensional problems generally involve considerably less data than two dimensional problems, and
3. the mathematical methods for handling two dimensional systems are generally less complete than those for one dimension.

As the techniques for processing multidimensional data have improved, the applications of digital signal processing have spread from one dimensional to two dimensional, to n-dimensional data. There exists many physical phenomena that inherently depend on two or more independent variables. In the prediction of weather and in seismic analysis, the data depends on more than one independent variable. Moreover, data originating from one dimensional processes can, in some cases, be considered two dimensional. For instance, data from periodic or cyclic processes can be represented as two dimensional arrays by using their periodicity.

Besides the general applicability of two dimensional signal processing in the above cases, other areas which have experienced significant growth in recent years include radar, sonar, and radio astronomy. The two dimensional processing of images is also a very important one. Images depend on two spatial variables, and are continuous in nature. However, if we digitize them and assume linear models in their formation, distortion, and recording we then have techniques that can be used in their processing. Depending on the applications, different processing techniques are used, notably: enhancement and restoration of images, and segmentation and encoding of images. For details, see References 1,2,3, 4 .

This brief discussion about applications of two dimensional digital signal processing shows that they are found in a wide variety of fields. In this thesis, we are interested in

extending one dimensional adaptive filtering techniques to two dimensions and then applying this in the area of image restoration. Once the transition from one to two dimensions is understood, the extension to n-dimensional signal processing is fairly straightforward.

A. OBJECTIVES OF THE THESIS

The use of the Wiener filter has proven to be a very powerful tool in the area of image restoration [Refs. 1,2, 4]. However, one disadvantage is the fact that the Wiener filter operates under the assumption that the image is stationary which generally is not the case in an image processing problem. In one dimensional signal processing the Wiener filter concept has provided a basis for various adaptive filtering algorithms. Within these adaptive algorithms, the filter possesses characteristics which can be modified to achieve some end or objective and is usually assumed to accomplish this "adaptation" automatically without the need for substantial intervention by the user. The adaptive filter can "learn" the signal characteristics when first turned on and thereafter can track changes in these characteristics. The first objective of this thesis is to examine a two dimensional Wiener model for image restoration which can then be extended to adaptive algorithms. Due to its relatively low computational requirements and the fact that it will work in a variety of signal environments, the least mean square (LMS) algorithm will be investigated first.

The second objective of this thesis is to develop a second two dimensional adaptive algorithm. In this case, we will work with the recursive least square (RLS) algorithm which offers faster convergence than the gradient-search-type algorithms but generally involves a greater cost per data sample and more numerical difficulties.

The final objective is to implement the LMS and RLS algorithms within a system identification model, a noise canceler, and an adaptive line enhancer. Each algorithm possesses several variants and various parameters which can be modified. A representative sample of the various outputs will be examined and the results compared in order to see which provides the optimum solution under given conditions.

B. ORGANIZATION OF THE THESIS

Chapter II is designed to review the development of the one dimensional Wiener filter and then extend it to two dimensions where it can be incorporated into a two dimensional LMS adaptive algorithm. Computer simulation of the algorithm within a system identification model is performed and the results are shown. The two dimensional RLS algorithm is derived in Chapter III and again the results of the algorithm

within a computer simulated system identification model are shown. Chapter IV contains the results of implementing the LMS and RLS algorithms in a noise canceler and an adaptive line enhancer. Conclusions concerning the results are also presented.

II. TWO DIMENSIONAL ADAPTIVE LEAST MEAN SQUARE ALGORITHM

In this chapter, we will review the derivation of the one dimensional Wiener filter and then develop an algorithm to extend it to the two dimensional case. We will use the two dimensional Wiener filter to achieve a two dimensional least mean square (LMS) adaptive filter algorithm which will be demonstrated to be useful in image processing by applying the algorithm in a noise canceler mode and in an adaptive line enhancer configuration for the restoration of images corrupted by noise.

A. ONE DIMENSIONAL WIENER FILTER

The problem of estimating one signal from another is one of the most important in signal processing. In many applications, the desired signal is not available or observable directly. Instead, the observable signal is a degraded or distorted version of the original signal. The signal estimation problem is to recover, in the best way possible, the desired signal from its degraded replica. One typical example which we will deal with in this paper is an image recorded by an imaging system that has been corrupted by noise. The problem is to undo the noise induced distortion and restore the original image.

This represents the classic one dimensional problem in communication theory where we must obtain an estimate of a signal of interest, which can be observed in the presence of some additive noise. In other words, the available information about the signal, $s(n)$, is contained in the received signal:

$$u(n) = s(n) + w(n) \tag{2.1}$$

where $w(n)$ is the noise. Therefore, we must process this available signal $u(n)$ through an optimal processor that produces the best possible estimate of $s(n)$.

In order to establish a two dimensional Wiener filter, we must first develop a one dimensional algorithm. This task has been approached from many different directions [Refs. 5,6,7]. The formulation by Haykin [Ref. 8] provides the most logical extension to two dimensions. First, we consider a tapped delay line filter similar to Figure 1 on page 5. The filter consists of a set of delay elements, a corresponding set of adjustable tap gains or coefficients $h(1), h(2),..., h(M)$ connected to the tap inputs, and a set of adders

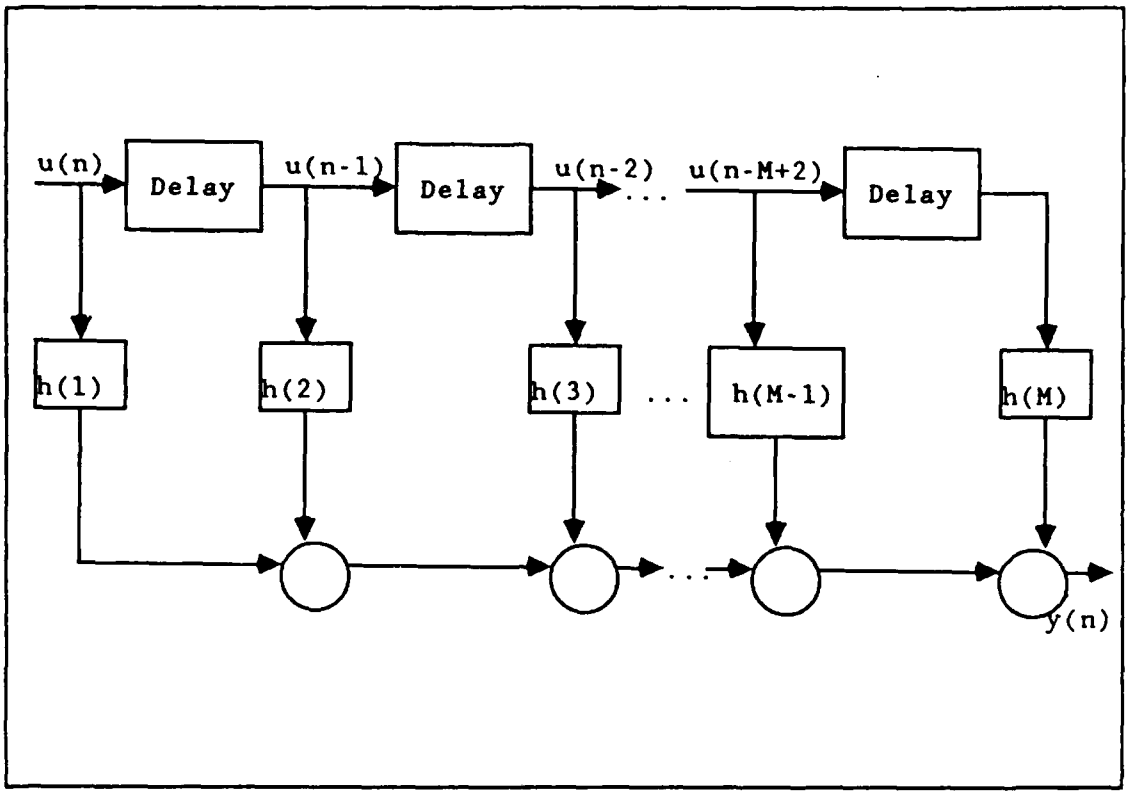

Figure 1. Tapped Delay Line Filter

for summing the resultant outputs. The filter is driven by a random time series producing the sequence $u(n)$, $u(n-1)$,..., $u(n-M+1)$ as the M tap inputs of the filter.

We can express the signal produced at the filter output, $y(n)$, by the convolution sum:

$$y(n) = \sum_{k=1}^{M} h(k)u(n-k+1). \tag{2.2}$$

We desire a filter which in some way minimizes the difference between some desired response, $d(n)$, and the corresponding value of the actual filter output. This difference can be denoted as

$$e(n) = d(n) - y(n) \tag{2.3}$$

where $e(n)$ is called the error signal. In Wiener theory, the filter is optimized by minimizing the mean-square value of this error signal, $e(n)$.

Let the mean-square value of the error be denoted by

$$MSE = E\{e^2(n)\} \tag{2.4}$$

where $E\{.\}$ is the expectation operator. This mean-square value is a real and positive scalar quantity, representing the average normalized power of the error signal, $e(n)$. Substituting Equation (2.3) into (2.4) yields

$$MSE = E\{d^2(n)\} - 2E\{d(n)y(n)\} + E\{y^2(n)\}. \tag{2.5}$$

Next, substituting Equation (2.2) into Equation (2.5) and then interchanging the orders of summation and expectation in the last two terms, we get

$$\begin{aligned} MSE = {} & E\{d^2(n)\} - 2\sum_{k=1}^{M} h(k)\, E\{d(n)u(n-k+1)\} \\ & + \sum_{k=1}^{M}\sum_{m=1}^{M} h(k)h(m)\, E\{u(n-k+1)u(n-m+1)\}. \end{aligned} \tag{2.6}$$

Assuming that the input signal u(n) and the desired response d(n) are jointly stationary, the three terms on the right-hand side of the above equation may be interpreted as follows:

1. The expectation $E\{d^2(n)\}$ is equal to the mean square value of the desired response d(n):

$$P_d = E\{d^2(n)\}. \tag{2.7}$$

2. The expectation $E\{d(n)u(n-k+1)\}$ is equal to the cross-correlation function of the desired response d(n) and the input signal u(n) for the lag of k-1. We can therefore write the single summation term on the righthand side of Equation (2.6) as follows:

$$\sum_{k=1}^{M} h(k)E\{d(n)u(n-k+1)\} = \sum_{k=1}^{M} h(k)p(k-1). \tag{2.8}$$

3. Finally, the expectation $E\{u(n-k+1)u(n-m+1)\}$ is equal to the autocorrelation function of the input signal u(n) for the lag of m-k:

$$r(m-k) = E\{u(n-k+1)u(n-m+1)\}. \tag{2.9}$$

Accordingly, we can rewrite the double summation term on the righthand side of Equation (2.6) in the form

$$\begin{aligned} & \sum_{k=1}^{M}\sum_{m=1}^{M} h(k)h(m)\, E\{u(n-k+1)u(n-m+1)\} \\ & \qquad = \sum_{k=1}^{M}\sum_{m=1}^{M} h(k)h(m)r(m-k). \end{aligned} \tag{2.10}$$

Thus, substituting Equations (2.7), (2.8), and (2.10) into Equation (2.6), we find that the expression for the mean square error may by rewritten in the form

$$MSE = P_d - 2\sum_{k=1}^{M} h(k)p(k-1) + \sum_{k=1}^{M}\sum_{m=1}^{M} h(k)h(m)r(m-k). \tag{2.11}$$

By differentiating Equation (2.11) with respect to h(k) and setting it equal to zero, we have the following set of M simultaneous equations

$$p(k-1) = \sum_{m=1}^{M} h_0(m)r(m-k), \quad k = 1,2,...,M. \tag{2.12}$$

This system of M simultaneous equations is called the normal equations with optimum filter coefficients as the unknowns. With the following definitions,

$$\mathbf{h}_0 = \begin{bmatrix} h_0(1) \\ h_0(2) \\ \cdot \\ \cdot \\ \cdot \\ h_0(M) \end{bmatrix} \tag{2.13}$$

$$\mathbf{p} = \begin{bmatrix} p(0) \\ p(1) \\ \cdot \\ \cdot \\ \cdot \\ p(M-1) \end{bmatrix} \tag{2.14}$$

$$\mathbf{R} = \begin{bmatrix} r(0) & r(1) & \cdots & r(M-1) \\ r(1) & r(0) & \cdots & r(M-2) \\ \cdot & \cdot & \cdots & \cdot \\ \cdot & \cdot & \cdots & \cdot \\ \cdot & \cdot & \cdots & \cdot \\ r(M-1) & r(M-2) & \cdots & r(0) \end{bmatrix} \tag{2.15}$$

we can rewrite the normal equations of Equation (2.12) in matrix form as

$$\mathbf{p} = \mathbf{R}\,\mathbf{h}_0 . \tag{2.16}$$

This represents the discrete-time version of the Wiener-Hopf equation.

B. TWO DIMENSIONAL WIENER FILTER

The following derivation parallels the development by Hadhoud and Thomas, however this research and simulation were completed separately and prior to the publication of Reference 9. In order to be applicable for an image processing problem, we must extend the formulation in the previous section to two dimensions [Refs. 10,11]. This is accomplished by developing a basic two dimensional Wiener filter as shown in Figure 2 on page 9. Within this filter, we use two input images, the reference array X and the primary input array D. The primary input image D is a two dimensional array which represents the ideal image plus additive noise, while the reference image X is noise that is assumed to be correlated to the noise in the primary input. Both the input arrays are MxM in dimension. The Wiener filter is an NxN causal FIR filter with a set of weights W_j defined as

$$W_j = \begin{bmatrix} W_j(0,0) & W_j(0,1) & \dots & W_j(0,N-1) \\ W_j(1,0) & W_j(1,1) & \dots & W_j(1,n-1) \\ . & . & \dots & . \\ . & . & \dots & . \\ W_j(N-1,0) & W_j(N-1,1) & \dots & W_j(N-1,N-1) \end{bmatrix} \tag{2.17}$$

which minimize the mean of the squared error, e_j, between the filter output and the desired input D. We designate j as the iteration number given by $j = mM + n$ where m and n take on the values from 0 to M-1. Just as in the one dimensional case, the filter output, $y(m,n)$, is the convolution sum of the filter mask and the reference input X which is given by

$$y(m,n) = \sum_{l=0}^{N-1} \sum_{k=0}^{N-1} W_j(l,k) X(m-l,n-k). \tag{2.18}$$

During the jth iteration the input from array X is represented by X_j where

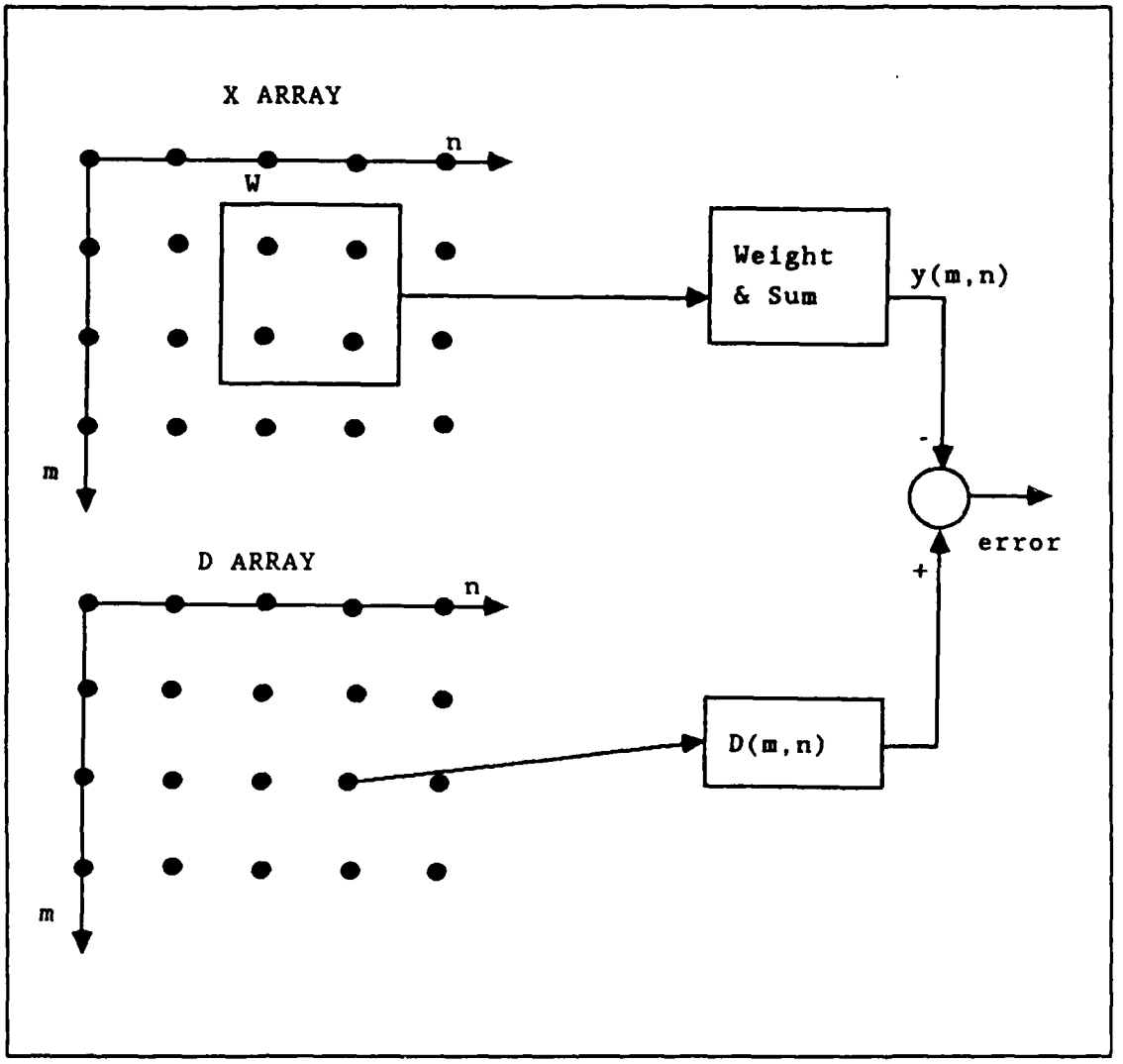

Figure 2. Two Dimensional Wiener Filter

$$X_j = \begin{bmatrix} X(m,n) & X(m,n-1) & \cdots & X(m,n-N+1) \\ \cdot & \cdot & \cdots & \cdot \\ \cdot & \cdot & \cdots & \cdot \\ \cdot & \cdot & \cdots & \cdot \\ X(m-N+1,n) & X(m-N+1,n-1) & \cdots & X(m-N+1,n-N+1) \end{bmatrix} \quad (2.19)$$

Using the Equations (2.17) and (2.19), Equation (2.18) can now be written as

$$y(m,n) = \sum_{l=0}^{N-1} \sum_{k=0}^{N-1} W_j(l,k)\, X_j(l,k) \tag{2.20}$$

for the *jth* iteration.

This output $y(m,n)$ can now be subtracted from one pixel D(m,n) of the array D to produce the error signal at the *jth* iteration

$$e_j = D(m,n) - \sum_{l=0}^{N-1} \sum_{k=0}^{N-1} W_j(l,k) X(m-l,n-k). \tag{2.21}$$

Since the purpose of the Wiener filter is to provide a set of weights which minimize the MSE, we can denote it as

$$MSE = E\{e_j^2\} \tag{2.22}$$

where $E\{.\}$ is the expectation operator.

Using a mathematical derivation similar to the one dimensional case of the previous section, we can see that

$$\begin{aligned} e_j^2 = D^2(m,n) - 2 \sum_{l=0}^{N-1} \sum_{k=0}^{N-1} W_j(l,k)\, D(m,n)\, X(m-l,n-k) \\ + \sum_{l=0}^{N-1} \sum_{k=0}^{N-1} \sum_{p=0}^{N-1} \sum_{q=0}^{N-1} W_j(l,k)\, W_j(p,q)\, X(m-l,n-k)\, X(m-p,n-q)\,. \end{aligned} \tag{2.23}$$

Substituting Equation (2.23) into Equation (2.22) yields

$$\begin{aligned} MSE = E[D^2(m,n)] - 2 \sum_{l=0}^{N-1} \sum_{k=0}^{N-1} W_j(l,k)\, E[D(m,n)\, X(m-l,n-k)] \\ + \sum_{l=0}^{N-1} \sum_{k=0}^{N-1} \sum_{p=0}^{N-1} \sum_{q=0}^{N-1} W_j(l,k)\, W_j(p,q)\, E[X(m-l,n-k)\, X(m-p,n-q)]\,. \end{aligned} \tag{2.24}$$

Defining P as the crosscorrelation matrix between the desired response D(m,n) and the reference input, R as the input autocorrelation matrix, and W as the optimum Wiener weight matrix, we can rewrite Equation (2.24) as

$$MSE = E[D^2(m,n)] - 2\sum_{l=0}^{N-1}\sum_{k=0}^{N-1} W_f(l,k)\,P(l,k) + \sum_{l=0}^{N-1}\sum_{k=0}^{N-1}\sum_{p=0}^{N-1}\sum_{q=0}^{N-1} W_f(l,k)\,W_f(p,q)\,R(p-l,q-k)\,. \tag{2.25}$$

Finally, if we minimize the MSE with respect to $W_f(l,k)$ then we have

$$P(l,k) = \sum_{p=0}^{N-1}\sum_{q=0}^{N-1} W_f(p,q)R(p-l,q-k). \tag{2.26}$$

This equation is the two dimensional equivalent to Equation (2.12). The matrix form of Equation (2.26) is given as

$$\mathbf{P} = \mathbf{R}\,\mathbf{W} \tag{2.27}$$

where the elements of this equation are defined as follows

$$\mathbf{R} = \begin{bmatrix} [R_0] & [R_1] & [R_2] & \cdots & \cdots \\ [R_1] & [R_0] & [R_1] & \cdots & \cdots \\ . & . & . & \cdots & \cdots \\ . & . & . & \cdots & \cdots \\ [R_{-N+1}] & [R_{-N+2}] & \cdots & \cdots & [R_0] \end{bmatrix} \tag{2.28}$$

$$\mathbf{W} = \begin{bmatrix} W_0 \\ W_1 \\ . \\ . \\ W_{N-1} \end{bmatrix} \tag{2.29}$$

$$\mathbf{P} = \begin{bmatrix} P_0 \\ P_1 \\ . \\ . \\ P_{N-1} \end{bmatrix} \tag{2.30}$$

Within the **R** matrix in equation (2.28) each element is a block Toeplitz matrix represented by the equation

$$R(p-l,q-k) = E[X(m-l,n-k)X(m-p,n-q)] \tag{2.31}$$

where l,k,p,q range from 0 to N-1.

C. TWO DIMENSIONAL LEAST MEAN SQUARE

One means of obtaining an approximate solution for the optimum weight matrix, **W**, is the use of a two dimensional LMS algorithm which is depicted in Figure 3 on page 13. This differs from Figure 2 on page 9 in that the error e_j is used to update the filter coefficients before shifting the data window X_j across the reference input for the next iteration. As in the one dimensional LMS algorithm, we are updating the coefficient matrix by adding the present matrix to a change proportional to the negative gradient of the error where the one dimensional instantaneous estimates of the gradient vector are based on sample values of the input and the error signal $e(n)u(n)$ [Ref. 8]. For the two dimensional *jth* iteration, we define the updated matrix as

$$W_{j+1} = W_j - \mu\, e_j\, X_j \tag{2.32}$$

where

W_{j+1}	updated weight matrix
W_j	previous weight matrix
μ	scaler multiplier controlling the rate of convergence and filter stability
$(e_j)(X_j)$	estimate for the 2-D instantaneous gradient

The previous equation can also be written as

$$W_{j+1}(l,k) = W_j(l,k) + 2\,\mu\,(e_j)\,X(m-l,n-k). \tag{2.33}$$

These two equations give the two dimensional weight updating algorithm for the LMS adaptive filter. The algorithm we have developed here may be implemented without any form of matrix operations, averaging, or differentiation.

The value of μ may be chosen based on the desired tracking ability, steady-state mean square error, and convergence speed. In signal processing, there are several methods for determining a suitable value, however in many of these cases it requires knowledge of the eigenvalues and eigenvectors of the autocorrleation matrix. In image

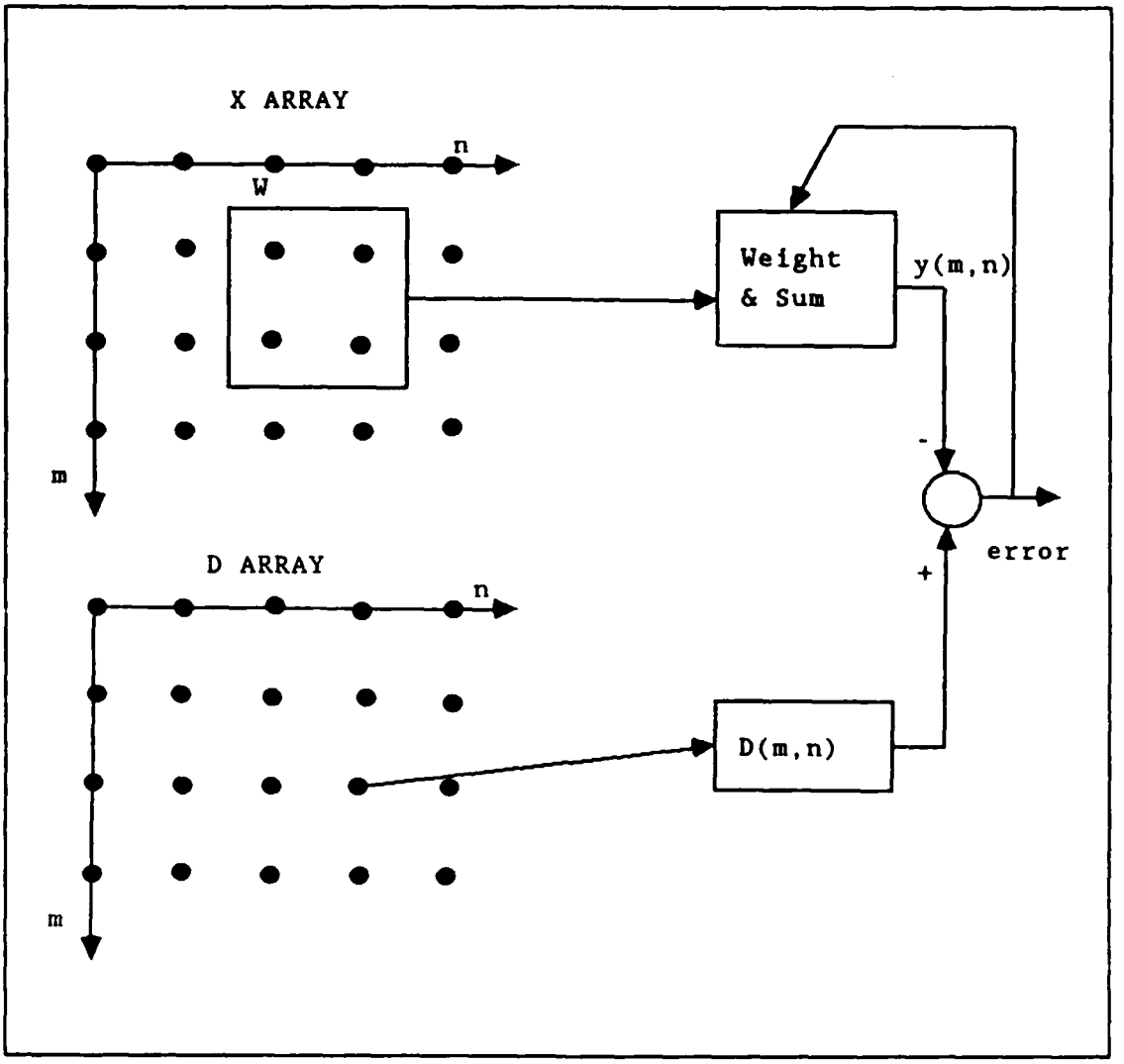

Figure 3. Two Dimensional Least Mean Square

processing, one method which does not require these values for the computation of μ is trial and error based on output image.

An alternate method used in one dimensional design which again does not require a priori knowledge of the autocorrelation matrix, discussed by Bitmead and Anderson [Ref. 12], is the normalized LMS. Using our previous notation, this method can be extended to two dimensions by first considering the two dimensional LMS update equation (2.32). In this equation, we redefine the step-size parameter, μ, for a given l and k as

$$\mu(l,k) = \frac{\alpha}{\beta + \sigma_j^2(l,k)} \tag{2.34}$$

in which α, represents the normalized step size chosen between zero and two, β is another small positive constant, and $\sigma_j^2(l,k)$ is one of the values from the matrix

$$\sigma_j^2 = \begin{bmatrix} \sigma_j^2(0,0) & \sigma_j^2(0,1) & \dots & \sigma_j^2(0,N-1) \\ \sigma_j^2(1,0) & \sigma_j^2(1,1) & \dots & \sigma_j^2(1,N-1) \\ . & . & \dots & . \\ . & . & \dots & . \\ \sigma_j^2(N-1,0) & \sigma_j^2(N-1,1) & \dots & \sigma_j^2(N-1,N-1) \end{bmatrix} \tag{2.35}$$

The element required from the matrix depends upon the current values of l and k being used for equation (2.33). The matrix may be be intitialized with the values of $(X(m-l,n-k))^2$ and to update the values in σ_j^2, we use the following equation

$$\sigma_{j+1}^2(l,k) = \rho(\sigma_j^2(l,k)) + (1-\rho)(X(m-l,n-k))^2 \tag{2.36}$$

where ρ is a weighting factor between zero and one. This ensures that the value of μ does not become large enough to cause the algorithm to become unstable.

D. IMPLEMENTATION

1. System Identification

In order to initially test the two dimensional least mean square algorithm, we established a system identification model shown in Figure 4 on page 15. Within this model the output of the known FIR filter, $d(m,n)$, is defined as

$$\begin{aligned} d(m,n) &= .4\,W(m,n) + .6\,W(m-1,n) \\ &- .3\,W(m,n-1) + .2\,W(m-1,n-1)\,. \end{aligned} \tag{2.37}$$

The output of the two dimensional least mean square filter, $y(m,n)$, consists of a set of adjustable coefficients and is defined as

$$\begin{aligned} y(m,n) &= A0\,W(m,n) + A1\,W(m-1,n) \\ &+ A2\,W(m,n-1) + A3\,W(m-1,n-1)\,. \end{aligned} \tag{2.38}$$

The adaptive filter output $y(m,n)$ is then compared with the known system output $d(m,n)$ to produce an error signal $e(m,n)$, defined as the difference between them.

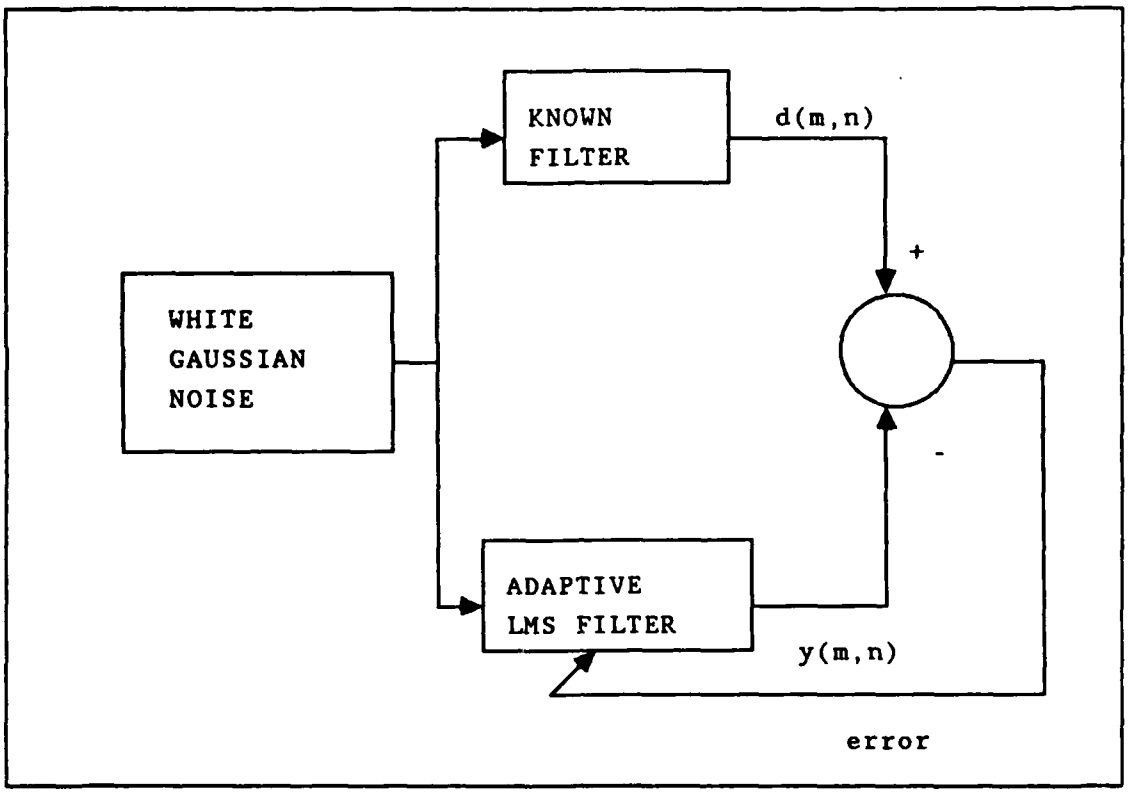

Figure 4. Two Dimensional System Identification

The operation of the adaptive filter is to minimize the error signal $e(m,n)$ by providing an adaptive process for adjusting the coefficients. For this adaptive process we use the update equation (2.32) for the two dimensional least mean square algorithm developed in the previous section

$$W_{j+1} = W_j - \mu e_j X_j \tag{2.32}$$

For this model a 32x32 white gaussian noise matrix was used as the input and the rate of convergence can be be seen in Figure 5 on page 16 and Figure 6 on page 17. Following 600 iterations all the coefficients had converged to within 10^{-3} of the actual coefficient value and the error was $3x10^{-5}$. A computer program for this system identification model is given in Appendix A.

2. Adaptive Noise Canceler

The usual method of estimating a signal corrupted by additive noise is to pass the composite signal through a filter that tends to suppress the noise while leaving the

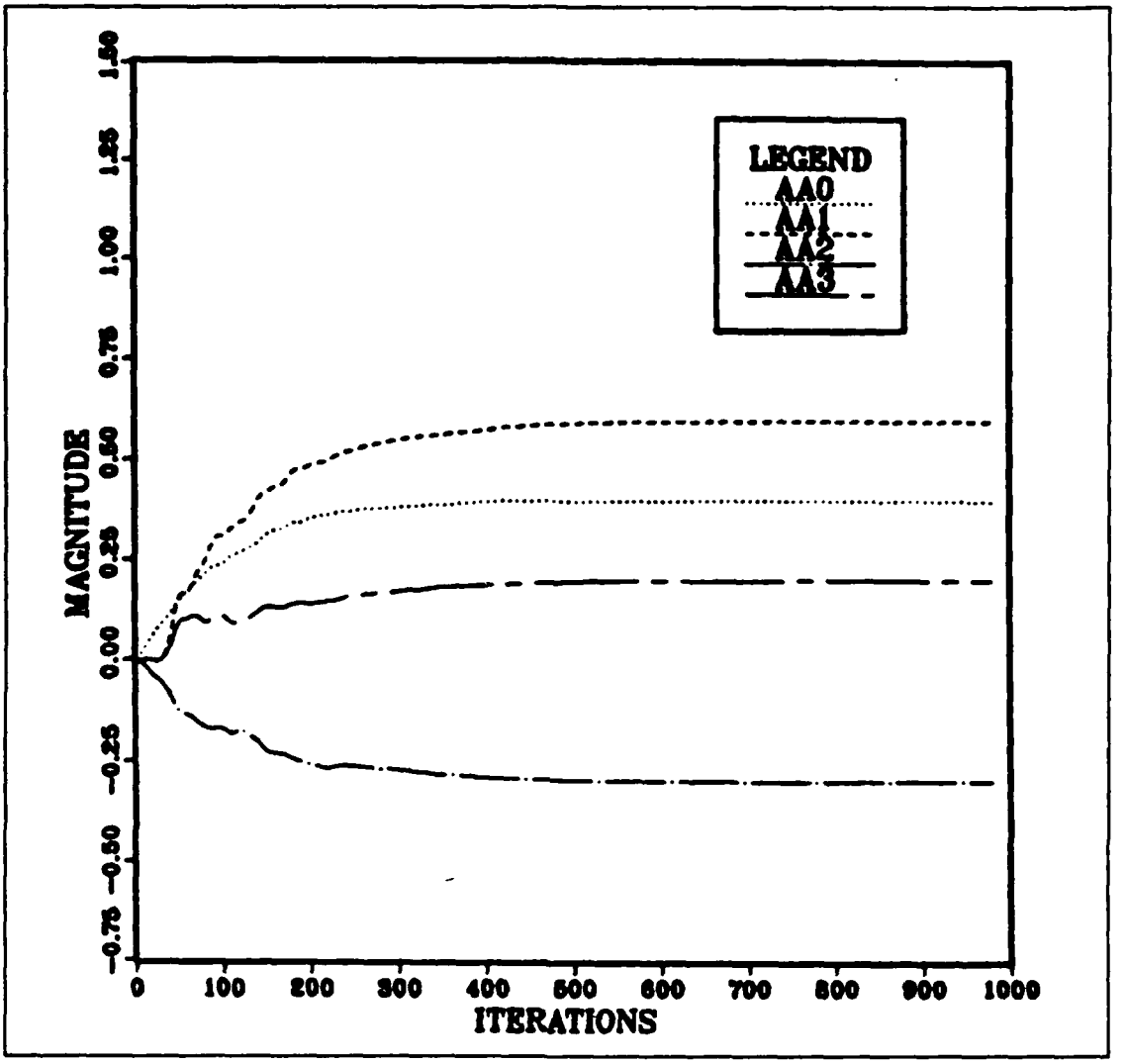

Figure 5. LMS System Identification-Rate of Convergence

signal relatively unchanged. The noise canceler and adaptive line enhancer developed by Widrow [Ref. 13] are well-documented ways of doing that. Adaptive noise canceling is a variation of optimal filtering that is highly advantageous in many applications. It uses an auxiliary or reference input derived from one or more sensors located at points in the noise field where the signal is weak or undetectable. This input is filtered and subtracted from a primary input containing both signal and noise. The reference input and the noise in the primary input are therefore correlated, and as a result the primary noise is attenuated or eliminated by cancellation.

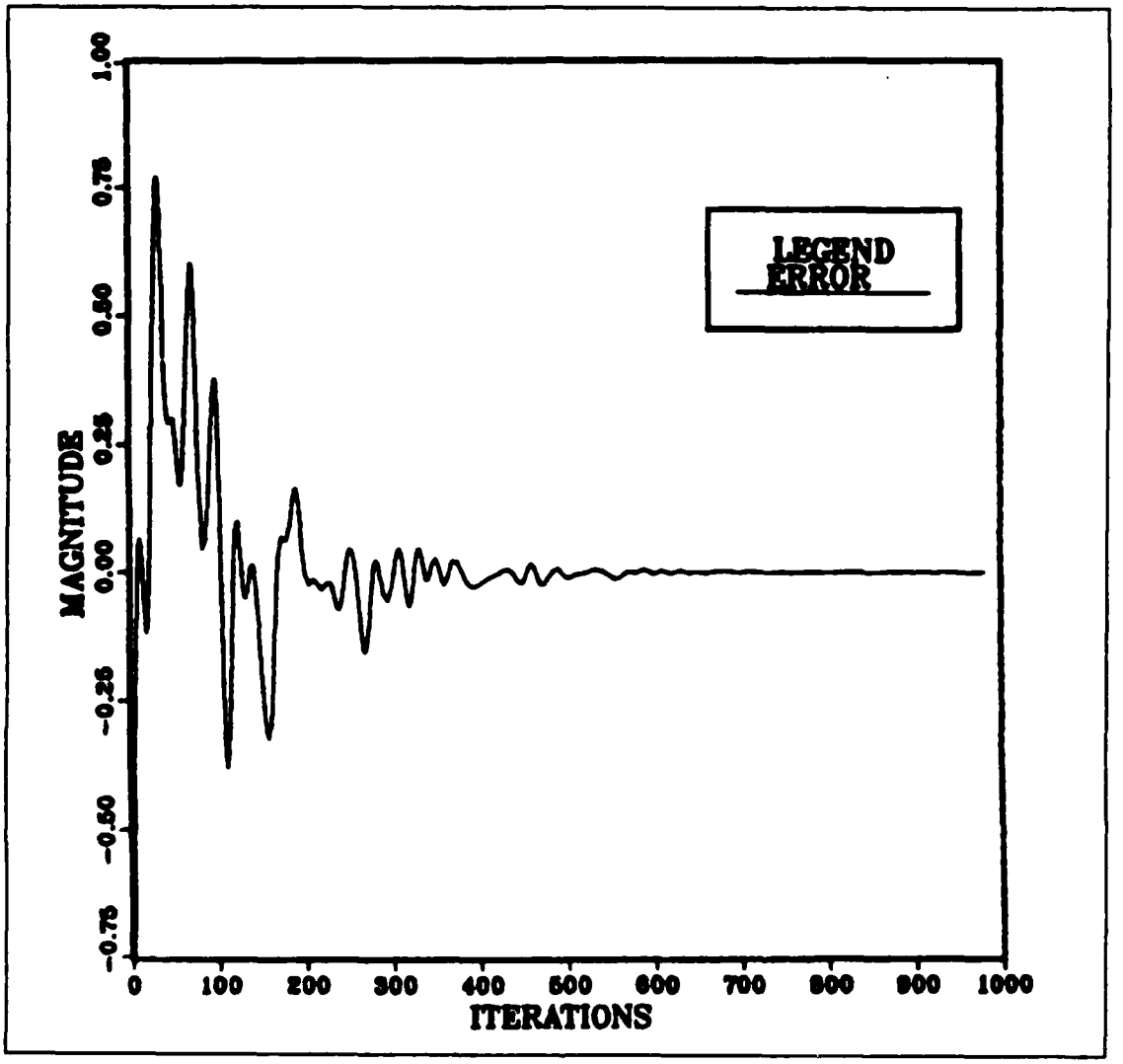

Figure 6. LMS System Identification-Rate of Convergence

The basic noise canceling concept is illustrated in Figure 7 on page 18. A signal is transmitted over a channel to a sensor that receives the signal plus noise, n_0. The combined signal and noise $s + n_0$ forms the "primary input" to the canceler. A second sensor receives a noise n_1 which is correlated in some way with the noise n_0. This sensor output provides the "reference input" to the canceler. The noise n_1 is filtered to produce an output y that is a close replica of n_0 . This output is subtracted from the primary input $s + n_0$ to produce the system output $s + n_0 - y$.

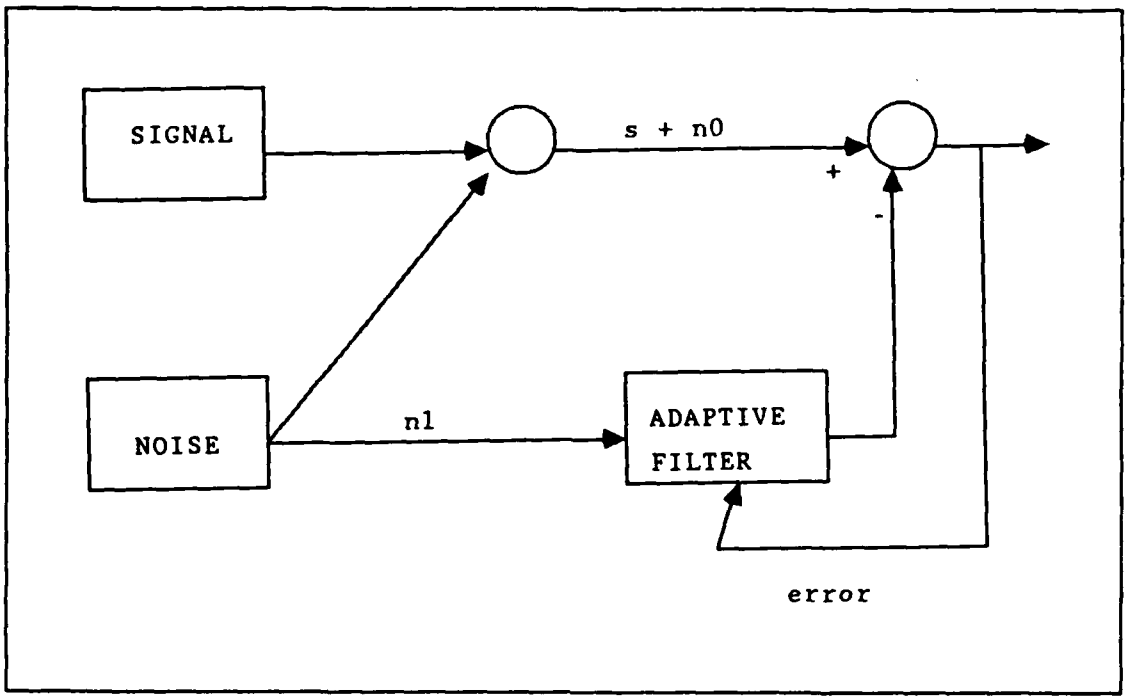

Figure 7. Adaptive Noise Canceler

In the system shown in Figure 7, the reference input is processed by our two dimensional least mean square filter. Thus the filter operates under changing conditions and will readjust itself continuously to minimize the error signal.

The computer program which simulates this noise canceling model is provided in Appendix B. We will discuss the results and conclusions concerning various simulations in Chapter 4.

3. Adaptive Line Enhancer

A special case of adaptive noise canceling is when there is only one signal $x(n)$ available which is contaminated by noise. In such a case, the signal $x(n)$ provides its own reference signal $y(n)$, which is taken to be a delayed replica of $x(n)$: $y(n) = x(n - \Delta)$, as shown in Figure 8 on page 19. The adaptive filter will respond by canceling any components of the main signal $x(n)$ that are in any way correlated with the secondary signal $y(n) = x(n - \Delta)$. Suppose the signal $x(n)$ consists of two components: a narrowband component that has long-range correlations and a broadband component which will tend to have short-range correlations. One of these could represent the desired signal and the other an undesired interfering noise. Suppose that the

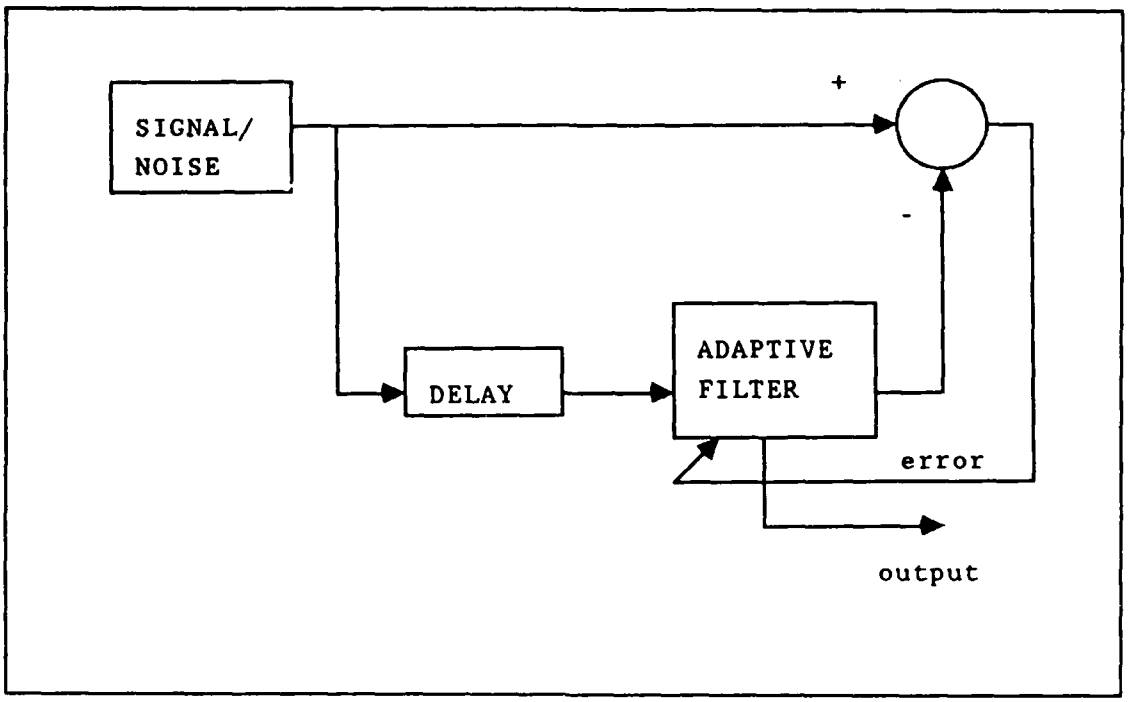

Figure 8. Adaptive Line Enhancer

delay Δ is selected so that it falls between the correlation lengths. Since Δ is longer than the effective correlation length of the broadband component, the delayed replica will be entirely uncorrelated with the broadband part of the main signal. The adaptive filter will not be able to respond to this component. On the other hand, since Δ is shorter than the correlation length of the narrowband component, the delayed replica that appears in the secondary input will be correlated with narrowband part of the main signal and the filter will respond to cancel it. Note that if Δ is selected to be longer than both correlation lengths, the secondary input will become uncorrelated with the primary input and the adaptive filter will turn itself off. In the opposite case, when the delay Δ is selected to be less than both correlation lengths, then both components of the secondary signal will be correlated with the primary signal, and therefore the adaptive filter will respond to cancel the primary $x(n)$ completely. The computational algorithm for the adaptive line enhancer is shown in the following three equations:

$$\hat{x}(n) = \sum_{m=0}^{M} h_m(n) y(n-m) = \sum_{m=0}^{M} h_m(n) x(n-m-\Delta) \tag{2.40}$$

$$e(n) = x(n) - \hat{x}(n) \tag{2.41}$$

$$h_m(n+1) = h_m(n) + 2\,\mu\, e(n)\, x(n-m-\Delta) \qquad m = 0,1,2,...,M \tag{2.42}$$

For this model we also developed a computer simulation which incorporates our two dimensional LMS algorithm and it is provided in Appendix C. The results and conclusions will again be discussed in Chapter 4.

III. TWO DIMENSIONAL ADAPTIVE RECURSIVE LEAST SQUARES ALGORITHM

In the previous chapter, under the LMS algorithm, the available data samples were used in order to attempt to move the current estimate of the impulse response to the optimum value. This approach has the advantage of being simple to implement but carries with it the disadvantages that it can be slow to approach the optimal weight vector and, once close to it, will usually fluctuate around the optimal vector rather than actually converge to it due to the effects of approximations made in the estimate of the performance function gradient.

To overcome these difficulties, we examine another approach in this chapter which uses the input data in such a way as to ensure optimality at each step. This alternative algorithm is based on the exact minimization of the least square criteria. This algorithm is known as recursive least squares (RLS).

A. ONE DIMENSIONAL RECURSIVE LEAST SQUARES

As in the LMS case, the one dimensional RLS algorithms is developed in several different ways [Refs. 5, 8]. Orfanidis [Ref. 6] provides a derivation which we will consider prior to our extension to two dimensions. The tapped delay line shown in Figure 9 on page 22 will provide the reference for the following discussion. We begin by replacing the LMS estimation criteria of $MSE = E[e^2]$ by

$$MSE = \sum_{k=0}^{n} e^2(k) \qquad k = 0, 1,...,n \tag{3.2}$$

where

$$e(k) = x(k) - \hat{x}(k) \tag{3.3}$$

and $\hat{x}(k)$ is the estimate of $x(k)$ produced by the Mth-order Wiener filter

$$\hat{x} = \sum_{m=0}^{M} h(m)\, y(k-m) \tag{3.4}$$

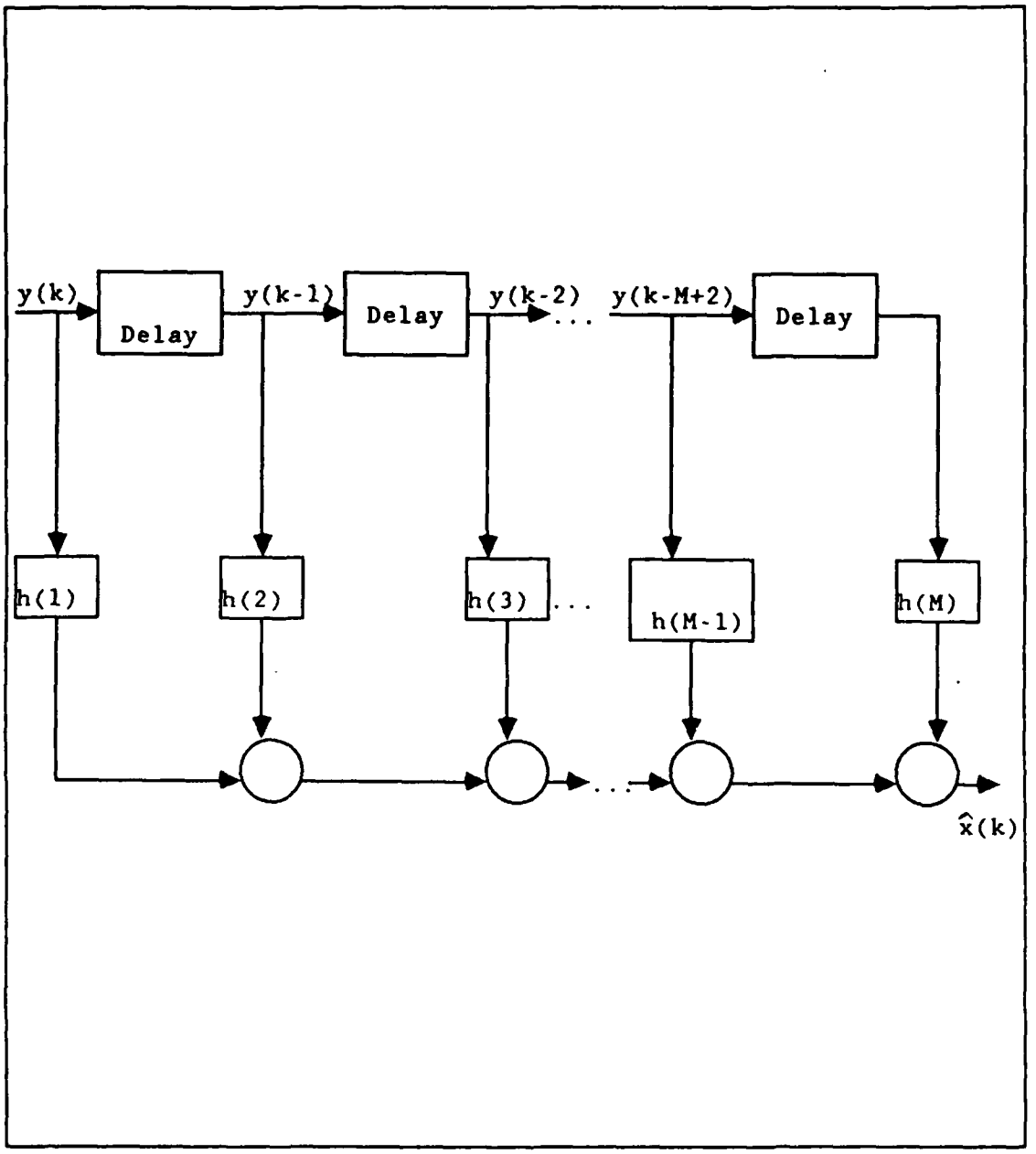

Figure 9. One Dimensional RLS Reference (TDL)

Substituting equation (3.2) into (3.1) and setting the derivative with respect to **h** to zero we find the least-squares analogs of the orthogonality equations

$$\frac{\partial MSE}{\partial \mathbf{h}} = -2\sum_{k=0}^{n} e(k)\,\mathbf{y}(k) = 0 \tag{3.5}$$

which may be rewritten in their normal equation form as follows

$$\sum_{k=0}^{n} (x(k) - \mathbf{h}^T\mathbf{y}(k))\mathbf{y}(k) = 0 \tag{3.6}$$

$$\left[\sum_{k=0}^{n} \mathbf{y}(k)\mathbf{y}(k)^T\right]\mathbf{h} = \sum_{k=0}^{n} x(k)\mathbf{y}(k) \tag{3.7}$$

Defining the quantities

$$R(n) = \sum_{k=0}^{n} \mathbf{y}(k)\,\mathbf{y}(k)^T \tag{3.8}$$

$$\mathbf{r}(n) = \sum_{k=0}^{n} x(k)\,\mathbf{y}(k) \tag{3.9}$$

we can then write the normal equation as $R(n)\,\mathbf{h} = \mathbf{r}(n)$, with solution $\mathbf{h} = R(n)^{-1}\,\mathbf{r}(n)$. Note that the n-dependence of $R(n)$ and $\mathbf{r}(n)$ depend on the current time n, therefore,

$$\mathbf{h}(n) = R(n)^{-1}\mathbf{r}(n) = P(n)\mathbf{r}(n) \tag{3.10}$$

where $P(n) = R(n)^{-1}$. These are the least squares analogs of the ordinary Wiener solution, with $R(n)$ and $\mathbf{r}(n)$ playing the role of the covariance matrix $E[\mathbf{y}(n)\mathbf{y}(n)^T]$ and cross-correlation vector $E[x(n)\mathbf{y}(n)]$, respectively. The RLS algorithm is obtained by writing equation (3.10) recursively in n and then using the following matrix inversion lemma [Ref. 5]

$$(A + BCD)^{-1} = A^{-1} - A^{-1}B(DA^{-1}B + C^{-1})^{-1}DA^{-1} \tag{3.11}$$

we get the update equation for the P matrix

$$P(n) = P(n-1) - \frac{P(n-1)\mathbf{y}(n)\mathbf{y}(n)^T P(n-1)}{1 + \mathbf{y}(n)^T P(n-1)\mathbf{y}(n)}. \tag{3.12}$$

Using the quantities in equations (3.8) and (3.9) to satisfy the recursions yields

$$R(n) = R(n-1) + \mathbf{y}(n)\mathbf{y}(n)^T \tag{3.13}$$

$$\mathbf{r}(n) = \mathbf{r}(n-1) + x(n)\mathbf{y}(n) \tag{3.14}$$

and substituting equations (3.12) and (3.14) into (3.10) after some mathematical manipulations we find

$$\mathbf{h}(n) = \mathbf{h}(n-1) + P(n)\,\mathbf{y}(n)\,e(n) \tag{3.15}$$

which differs from the LMS alogorithm by the presence of the $P(n)$, vice μ, in front of the weight correction term. Since $P(n) = R(n)^{-1}$ is a measure of the covariance matrix $E[\mathbf{y}(n)\mathbf{y}(n)^T]$, the presence of $R(n)^{-1}$ makes the RLS algorithm behave like Newton's method, and hence has fast convergence properties.

B. TWO DIMENSIONAL RECURSIVE LEAST SQUARES

Although it is discussed briefly by Wellstead and Caldas Pintos [Ref. 14], limited information is available in the open literature in this area. In order to develop a two dimensional recursive least square (RLS) algorithm we will extend the one dimensional adaptive algorithm developed in the previous section in a method similar to that used for the LMS algorithm.

As in Chapter 2, using Figure 10 on page 25 as a reference we see that the basic filter has two input images. The primary two dimensional input array, D, is the ideal image plus additive noise. The reference image X is the noise array. Each array is of dimension M by M. The filter mask, W, is N by N.

The one dimensional RLS algorithm, equation (3.15), is the same as the one dimensional LMS algorithm with the exception that $P(n)$ replaces μ. Therefore, if a method of developing a P matrix for the two dimensional case can be devised we can then use an algorithm similar to the two dimensional LMS algorithm to update the filter coefficients. First, let

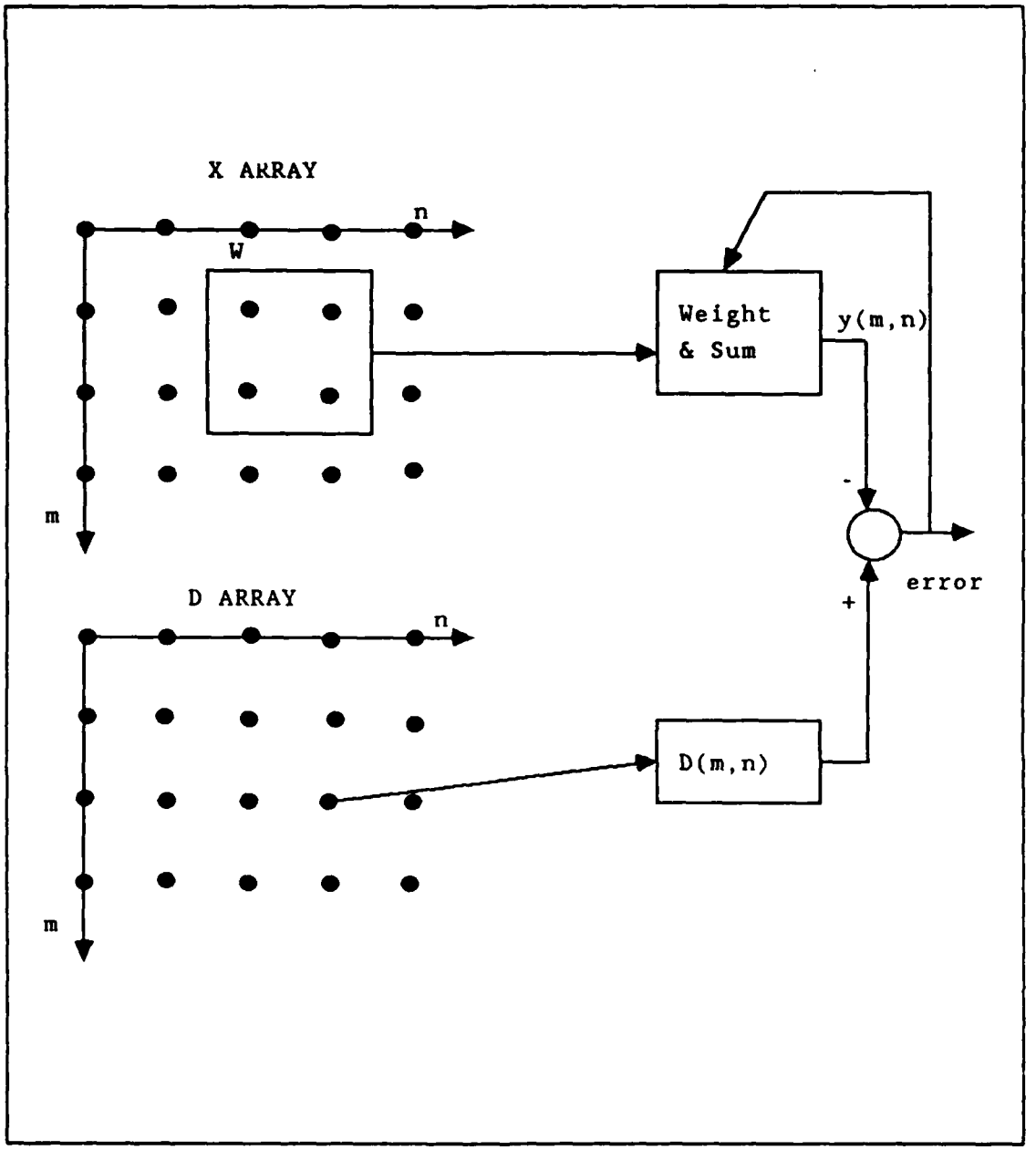

Figure 10. Two Dimensional RLS Reference

$$P_j = \begin{bmatrix} P_j(0,0) & P_j(0,1) & \cdots & P_j(0,N^2-1) \\ \cdot & \cdot & \cdots & \cdot \\ \cdot & \cdot & \cdots & \cdot \\ \cdot & \cdot & \cdots & \cdot \\ P_j(N^2-1,0) & P_j(N^2-1,1) & \cdots & P_j(N^2-1,N^2-1) \end{bmatrix} \tag{3.16}$$

which represents the P matrix on the jth iteration where j is defined as $j = mM + n$. As in the two dimensional LMS algorithm, we let the filter coefficient matrix be given as

$$W_j = \begin{bmatrix} W_j(0,0) & W_j(0,1) & \cdots & W_j(0,N-1) \\ W_j(1,0) & W_j(1,1) & \cdots & W_j(1,N-1) \\ \cdot & \cdot & \cdots & \cdot \\ \cdot & \cdot & \cdots & \cdot \\ \cdot & \cdot & \cdots & \cdot \\ W_j(N-1,0) & W_j(N-1,1) & \cdots & W_j(N-1,N-1) \end{bmatrix} \tag{3.17}$$

and the input data window as

$$X_j = \begin{bmatrix} X(m,n) & X(m,n-1) & \cdots & X(m,n-N+1) \\ \cdot & \cdot & \cdots & \cdot \\ \cdot & \cdot & \cdots & \cdot \\ \cdot & \cdot & \cdots & \cdot \\ X(m-N+1,n) & X(m-N+1,n-1) & \cdots & X(m-N+1,n-N+1) \end{bmatrix} \tag{3.18}$$

In order to establish an update equation for the P matrix in which each element will have the proper dimensions, we use equations (3.17) and (3.18) and make the following transformations; the filter mask, W_j, equation (3.17) is transformed into a vector defined as

$$WW_j = \begin{bmatrix} W_j(0,0) \\ W_j(0,1) \\ \cdot \\ \cdot \\ W_j(0,N-1) \\ W_j(1,0) \\ W_j(1,1) \\ \cdot \\ \cdot \\ W_j(1,N-1) \\ \cdot \\ \cdot \\ W_j(N-1,0) \\ W_j(N-1,1) \\ \cdot \\ \cdot \\ W_j(N-1,N-1) \end{bmatrix} \tag{3.19}$$

and we transform the X_j matrix, equation (3.18) into

$$XX_j = \begin{bmatrix} X(m,n) \\ X(m,n-1) \\ \cdot \\ X(m,n-N+1) \\ \cdot \\ \cdot \\ X(m-N+1,n) \\ X(m-N+1,n-1) \\ \cdot \\ X(m-N+1,n-N+1) \end{bmatrix} \tag{3.20}$$

Using the quanitities defined above, we can then write the P matrix update equation as

$$P_j = P_{j-1} - \frac{P_{j-1}(XX_j)(XX_j^T)P_{j-1}}{1 + (XX_j^T)P_{j-1}(XX_j)} \tag{3.21}$$

As discussed earlier, the one dimensional RLS weight updating algorithm differs from the LMS in that it contains the P matrix while the LMS algorithm contains μ. This must be considered in two dimensions since P is a matrix and μ is a scalar. Therefore using the previous equations, we can obtain the update to the filter coefficients via

$$WW_{j+1} = WW_j - (P_j)(e_j)(XX_j) \tag{3.22}$$

where e_j is given as

$$e_j = D(m,n) - y(m,n) \tag{3.23}$$

$y(m,n)$ is the convolution sum of the image in the reference input with the filter window

$$y(m,n) = \sum_{l=0}^{N-1} \sum_{k=0}^{N-1} W_j(l,k)\, X_j(l,k) \tag{3.24}$$

and the value P_j is updated by equation (3.21).

The cost function of the RLS criterion could be modified to include a windowing of the input data as follows

$$MSE = \sum_{k=0}^{n} q^{n-k} e^2(k) \tag{3.25}$$

This modification results in the following modified P matrix update:

$$P_j = \frac{1}{q}\left[P_{j-1} - \frac{P_{j-1}(XX_j)(XX_j^T)P_{j-1}}{1 + (XX_j^T)P_{j-1}(XX_j)} \right] \tag{3.26}$$

where q, the averaging or "forgetting factor" is a positive constant. It is usually chosen to be slightly less than one, thereby diminishing the contribution of the "older" data. The problem of data nonstationarity is the main reason for introducing this type of weighting factor.[Ref. 6] It should be noted that the usual RLS algorithm is attained when $q = 1$ and that no changes in the amount of computation is required.

C. IMPLEMENTATION

1. System Identification

Following the mathematical development of the two dimensional recursive least squares algorithm, we implemented and tested it as was previously done with the two dimensional least mean square algorithm. Our initial testing was done in a system identification model as shown in Figure 11. As in the two dimensional LMS case, we used a known filter with the following output equation

$$d(m,n) = .4W(m,n) + .6W(m-1,n) - .3W(m,n-1) + .2W(m-1,n-1) \tag{3.27}$$

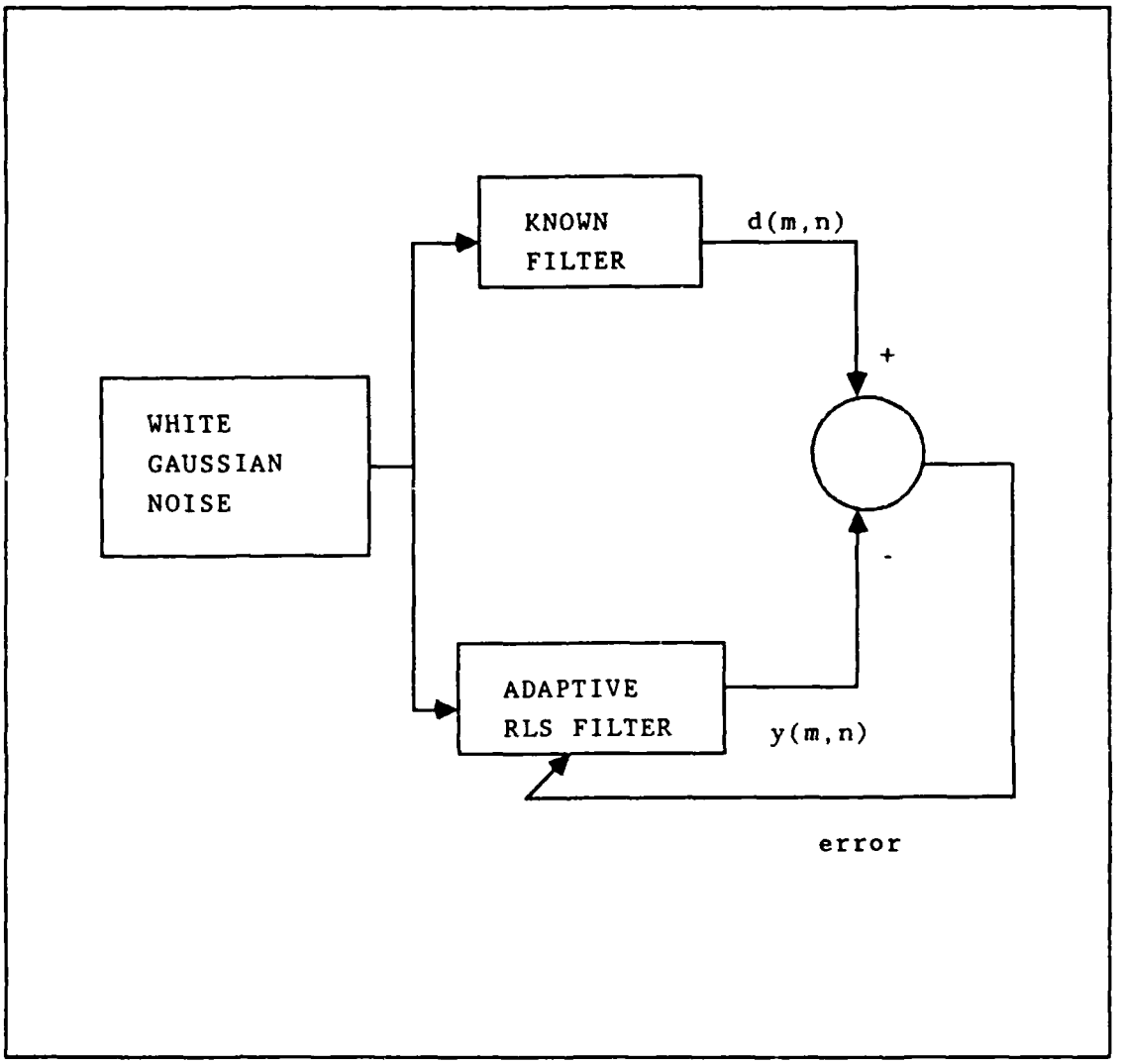

Figure 11. Two Dimensional RLS System Identification

The two dimensional adaptive recursive least square filter output was given as

$$y(m,n) = A0\, W(m,n) + A1\, W(m-1,n) + A2\, W(m,n-1) + A3\, W(m-1,n-1) \tag{3.28}$$

The error signal was generated by taking the difference between the desired (known) signal, $d(m,n)$, and the filter output, $y(m,n)$. By using equations (3.21) and (3.22) to update the P matrix and the filter coefficients, we caused the filter weights to move toward their optimum values and the error signal to approach zero.

The computer program for this model is provided is Appendix D. For this system, a 32x32 white gaussian noise matrix was generated for the input to both the known filter and the RLS adaptive filter. The rate of convergence is shown in Figure 12 on page 31 and Figure 13 on page 32 and as can be seen after approximately 70 iterations each of the coefficients had converged to with 10^{-3} of the actual value and the error was 10^{-5}.

2. Adaptive Noise Canceler/Adaptive Line Enhancer

These two systems were both discussed in Chapter 2 with regard to implementation of the two dimensional LMS algorithm. In order to further test the RLS algorithm we also implemented it within the noise canceler and the adaptive line enhancer. The computer programs which were used for the simulation are given Appendix E and Appendix F, respectively. The simulation results and discussion will be provided in Chapter 4.

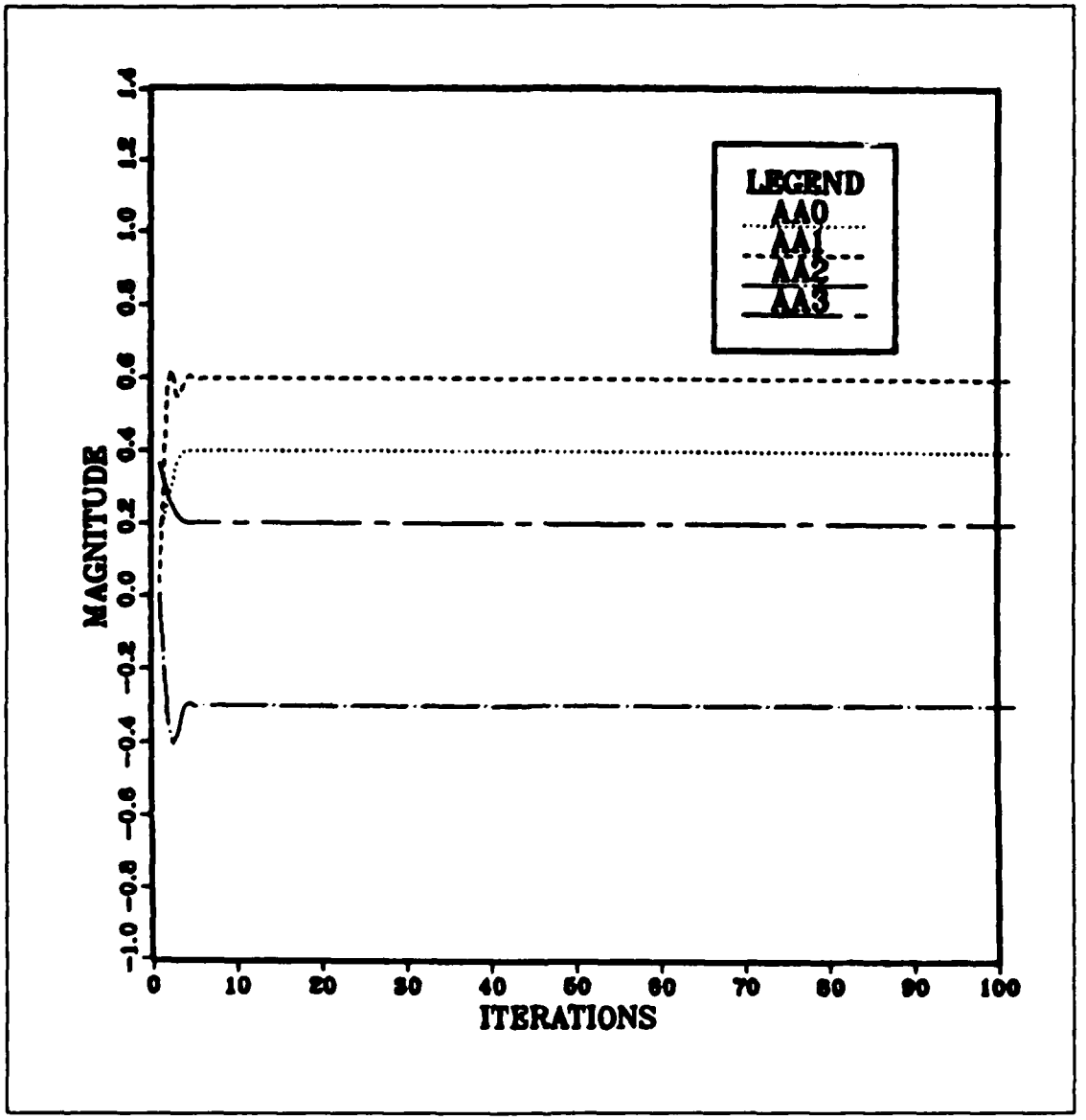

Figure 12. RLS System Identification Rate of Convergence

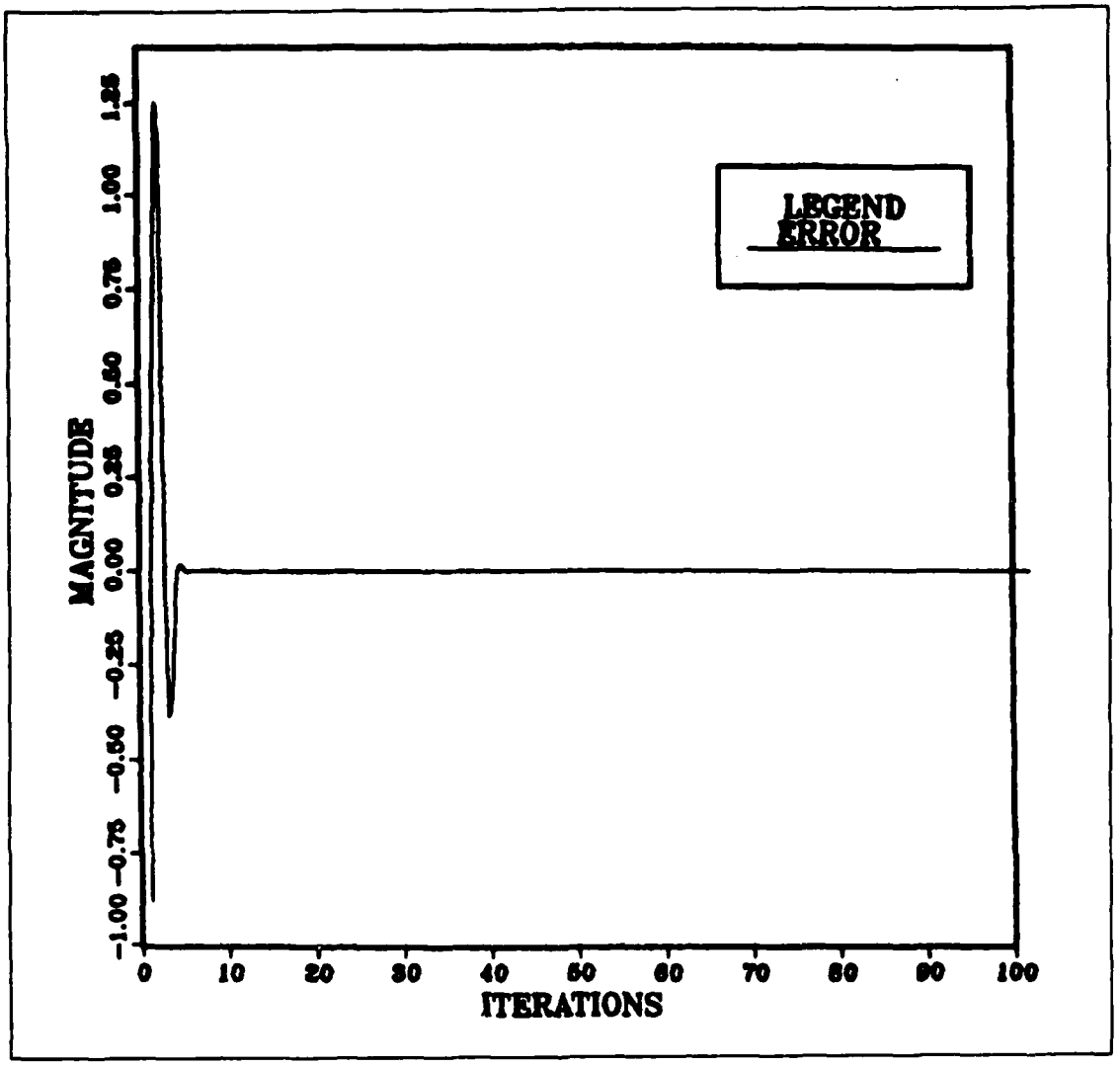

Figure 13. RLS System Identification Rate of Convergence

IV. RESULTS AND CONCLUSIONS

In this chapter, we present the experimental results from our implementation of the two dimensional LMS algorithm derived in Chapter II and the two dimensional RLS algorithm derived in Chapter III. The two algorithms were used in a noise canceler and an adaptive line enhancer, as discussed in Chapter II, to solve an image restoration problem.

Figure 14 shows the original image of a house which is comprised of 128 by 128 eight bit pixels. The image has a mean value of 131.68 and a variance of 3194.4. In Figure 15 on page 34, we have corrupted the original image with additive white gaussian noise (zero mean and variance 1600).

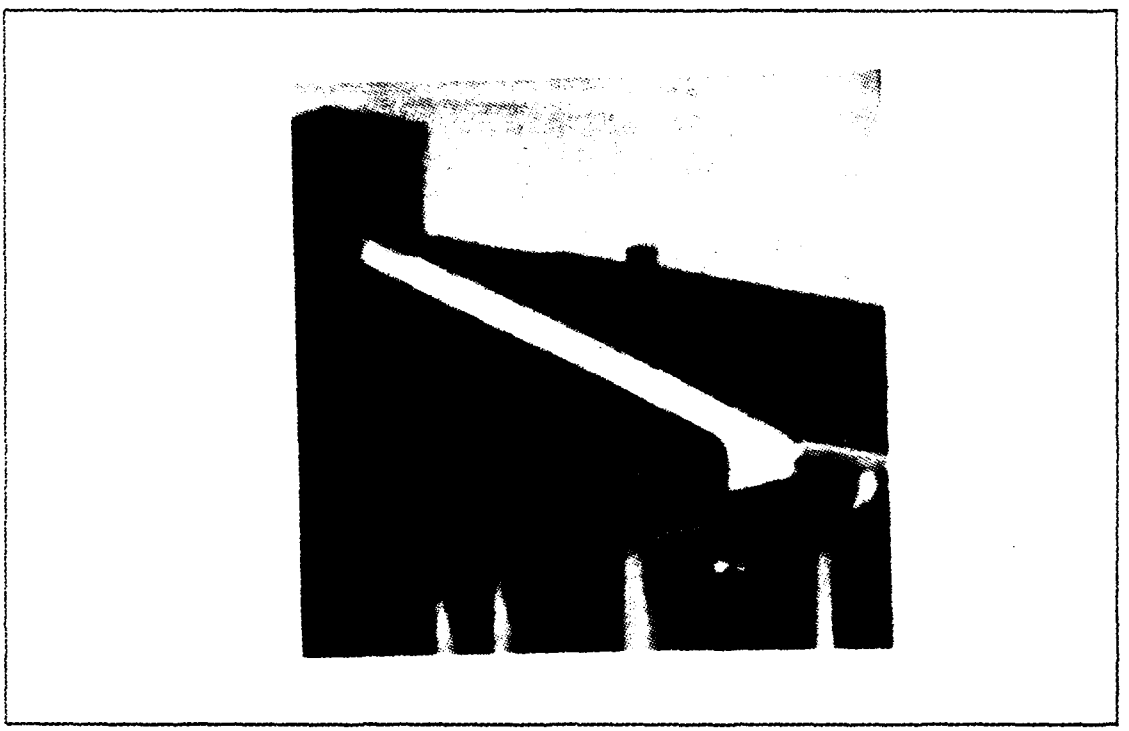

Figure 14. Original Image

Figure 15. Image plus Additive Noise

A. NOISE CANCELER RESULTS

In order to solve our image restoration problem, we initially implemented both the LMS and the RLS algorithms in a noise canceler using a two by two filter matrix and a value of $70x10^{-9}$ for μ in the LMS algorithm.

Figure 16 on page 35 shows the output when the LMS algorithm is processed through the image for one pass; this results in 16,384 updates of the filter corresponding to the 16,384 pixels. Figure 17 on page 36 is the results following two passes through the image. No significant improvement was realized with more than two passes.

Implementing a two by two RLS adaptive filter in the noise canceler produced very favorable results after one pass through the image. These results are shown in Figure 18 on page 36 and compare well with the original image and the LMS output after two passes.

For the LMS algorithm our best results were achieved with a value of $70x10^{-9}$ for μ, however in order to demonstrate the effect of changing this value Figure 19 on page 37 represents the output with $\mu = 35x10^{-9}$ and Figure 20 on page 37 is produced by a spatially varying normalized μ as discussed in Chapter II. It can be seen that for various

values of μ, different features within the image were restored at different levels. In Figure 16 on page 35, the edges and more detailed segments of the image were restored better than in Figure 19 on page 37, however areas of similar contrast were not as well restored. When using the normalized μ the mean value more closely approached the original mean, however the variance was reduced.

Finally, increasing the number of coefficients, using a three by three filter matrix in the LMS algorithm we obtained the results shown in Figure 21 on page 38 in one pass. This filter showed no significant improvement in the variance compared to the two by two filter; however, it did improve with respect to the mean.

Table 1 on page 38 shows the restoration results comparing the mean and variance of the various outputs with that of the original image and the image plus noise.

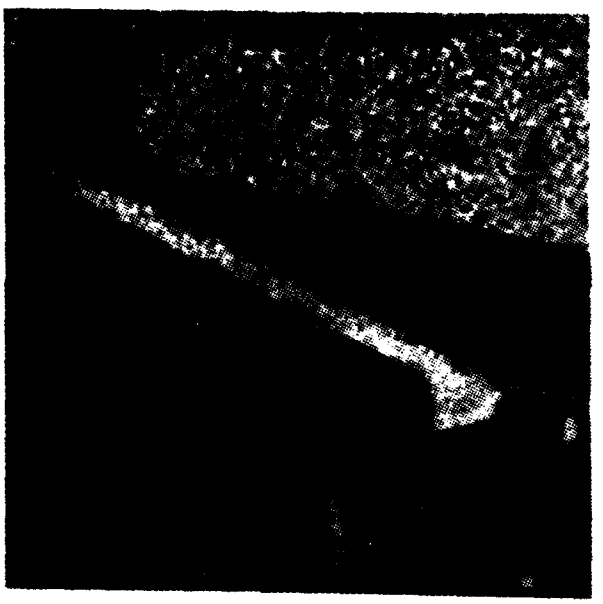

Figure 16. Restored Image: Noise Canceler/LMS (One pass)

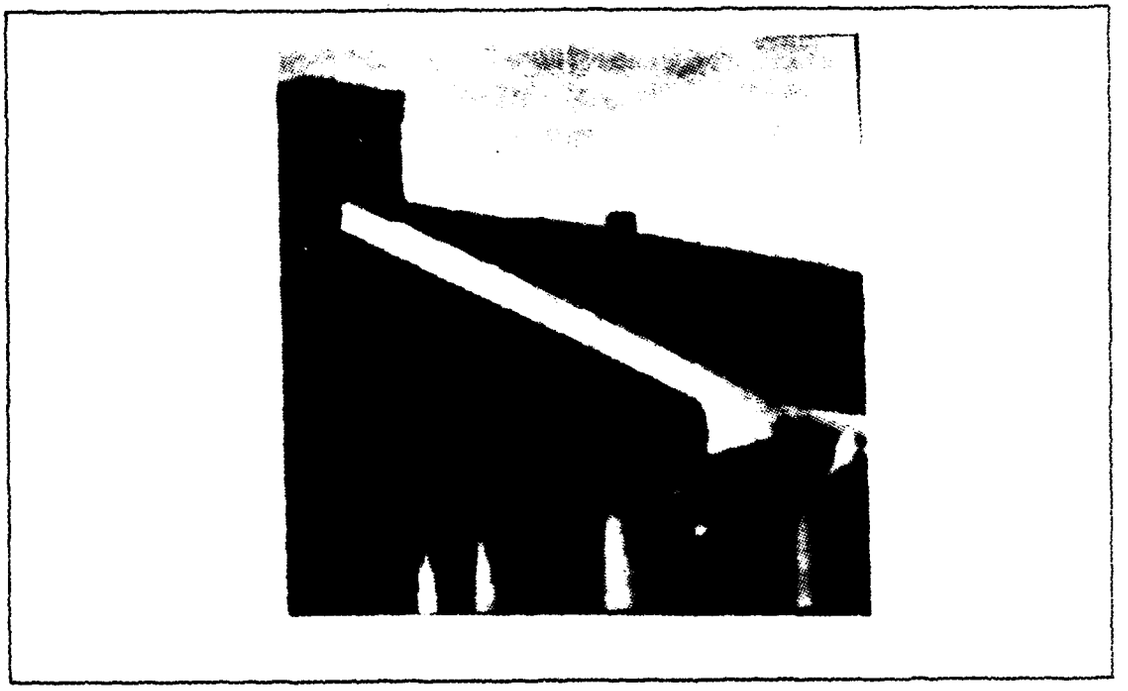

Figure 17. Restored Image: Noise Canceler/LMS (Two passes)

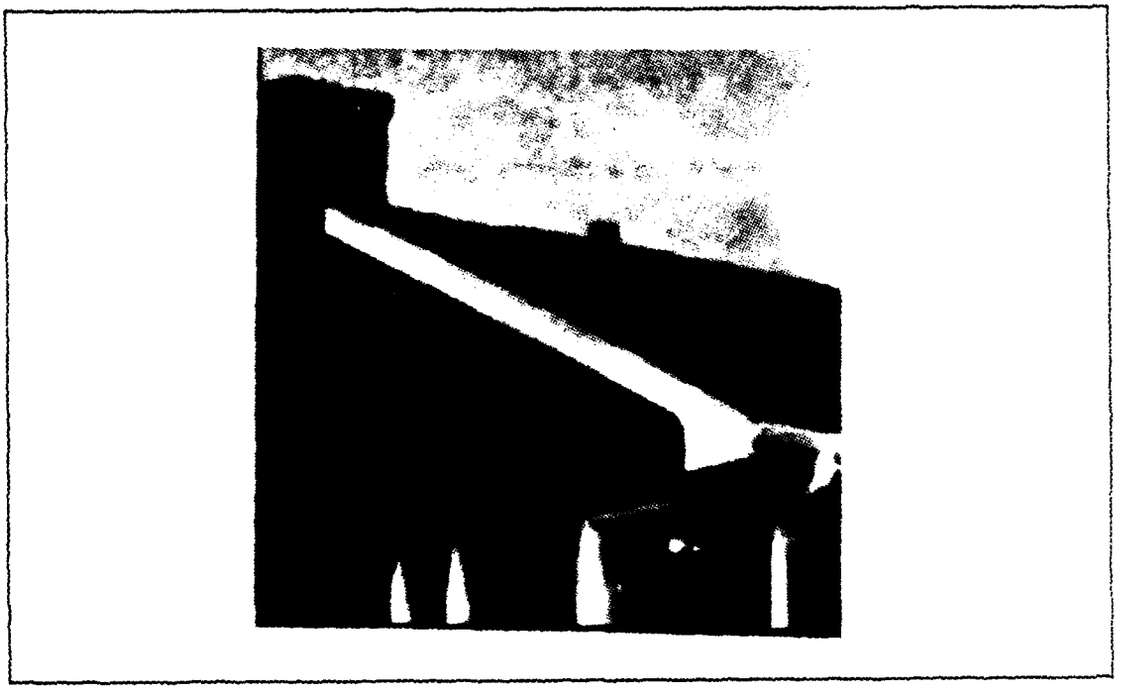

Figure 18. Restored Image: Noise Canceler/RLS (One pass)

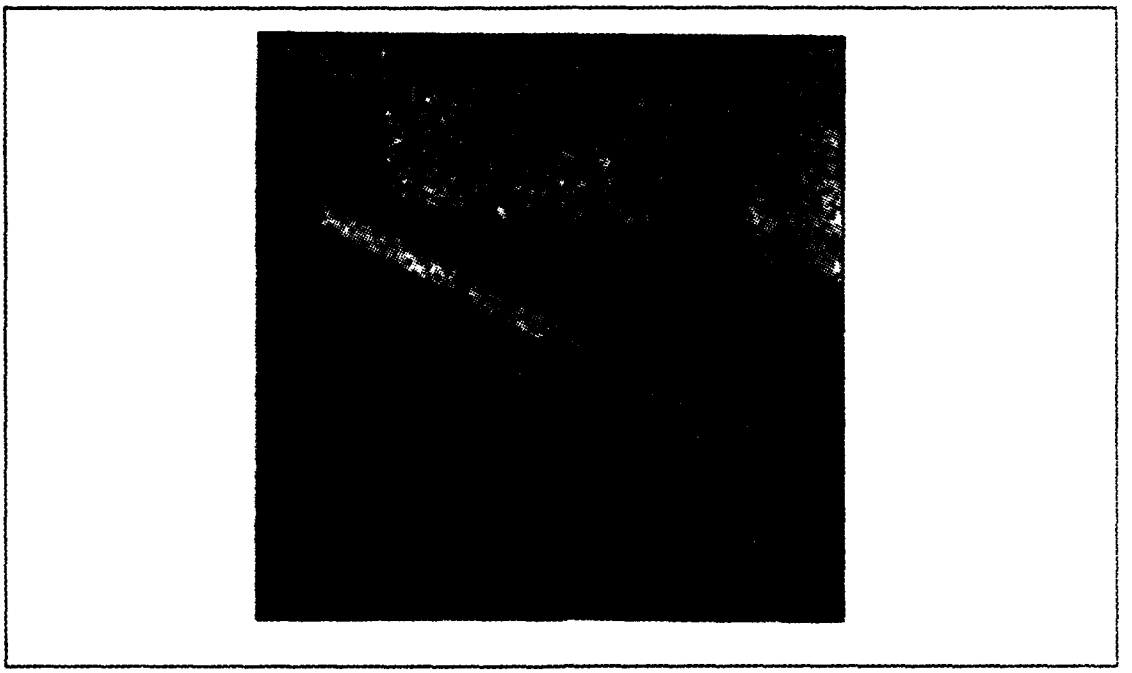

Figure 19. Restored Image: Noise Canceler/LMS (different mu)

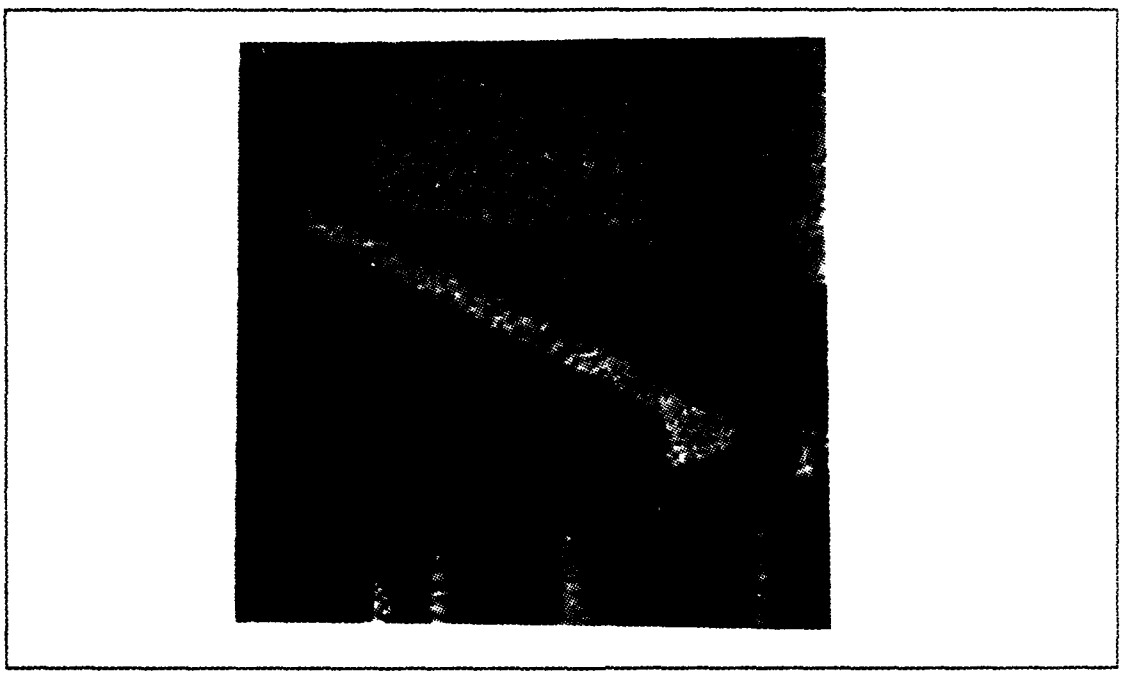

Figure 20. Restored Image: Noise Canceler/LMS (normalized mu)

Figure 21. Restored Image: Noise Canceler/LMS (3 by 3)

Table 1. NOISE CANCELER RESULTS

TYPE OF FILTER	MEAN	VARIANCE
Original Image	131.68	3194.4
Image plus Noise	134.25	1647.7
One pass 2x2 LMS	114.52	2034.3
Two pass 2x2 LMS	136.78	3711.8
One pass 2x2 RLS	139.04	3607.7
One pass LMS(different mu)	120.12	1940.4
One pass LMS(normalized mu)	126.53	1500.7
One pass 3x3 LMS	119.13	2017.3

B. ADAPTIVE LINE ENHANCER RESULTS

We next implemented our two algorithms in an adaptive line enhancer. In order to obtain a reference input from the primary input, we used a two dimensional delay operator of (m-1,n). As in the noise canceler we again used a two by two filter matrix and this produced the results shown in Figure 22 on page 39 and Figure 23 on page 40 for the one pass LMS and the one pass RLS, respectively. No significant improvement was noted in the two pass LMS compared to the one pass LMS.

Table 2 on page 40 shows the comparison of each output mean and variance with the mean and variance of the original image and the image plus noise.

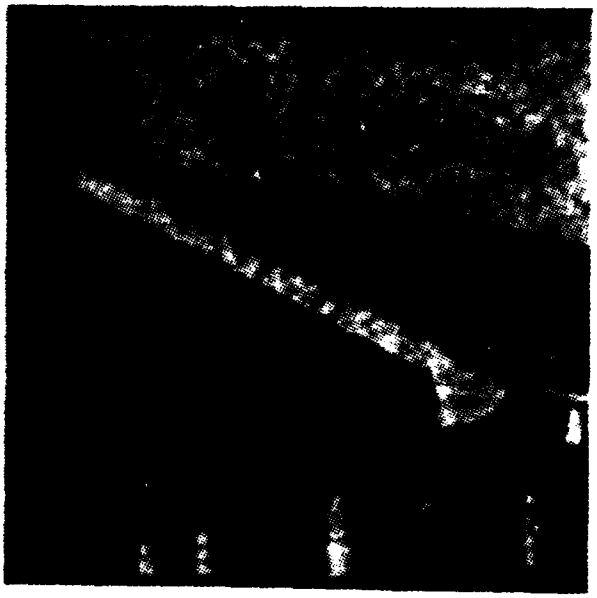

Figure 22. Restored Image: Adaptive Line Enhancer/LMS (One pass)

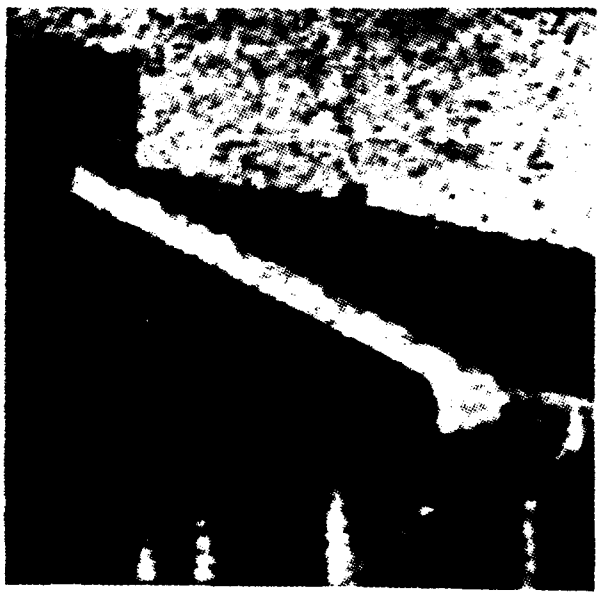

Figure 23. Restored Image: Adaptive Line Enhancer/RLS (One pass)

Table 2. ADAPTIVE LINE ENHANCER RESULTS

TYPE OF FILTER	MEAN	VARIANCE
Original Image	131.68	3194.4
Image plus Noise	134.25	1647.7
One pass 2x2 LMS	125.03	2521.8
Two pass 2x2 LMS	126.94	2529.9
One pass 2x2 RLS	139.69	2124.2

C. CONCLUSIONS

In this thesis we have extended a one dimensional LMS and a one dimensional RLS adaptive algorithm to two dimensions. As we examine the results we see that many of the one dimensional comparisons between LMS and RLS are applicable to two dimensions:

1. the rate of convergence of the RLS algorithm is in general faster than that of the LMS algorithm by an order of magnitude
2. this superior performance of the RLS algorithm is attained at the expense of a large increase in the computational complexity

3. there are no approximations within the derivation of the RLS algorithm, therefore as the number of iterations approach infinity, the least squares estimate approaches the optimum Wiener value
4. in the RLS algorithm the correction which is applied is computed using all past data whereas the LMS algorithm uses only the instantaneous sample and the error signal; this is not necessarily an advantage in image processing.

The objectives of this thesis were successfully completed. Some suggestions for future work include: (i) to examine these two dimensional algorithms in other applications, applying them to areas other than image processing, (ii) to extend the two dimensional LMS and RLS algorithms to n-dimensions and implement them, (iii) to analyze the extension of these one dimensional algorithms to two dimensions in the frequency domain. Multidimensional digital signal processing is being applied in many areas today, however the potential for future growth and applications appears unlimited.

APPENDIX A. LMS SYSTEM IDENTIFICATION PROGRAM

```
C     SYSTEM IDENTIFICATION OF AN ADAPTIVE TWO DIMENSIONAL
C         LMS FILTER VS. A KNOWN TWO DIMENSIONAL FILTER
C
      REAL MU
      DIMENSION A0(0:32,0:32),A1(0:32,0:32),A2(0:32,0:32),
     *  A3(0:32,0:32),E(0:32,0:32),D(0:32,0:32),W(-1:32,-1:32),
     *  Y(0:32,0:32)
C
C     ******VARIABLE DEFINITIONS******
C     A=NOISE AMPLITUDE
C     S=NOISE VARIANCE
C     AM=NOISE MEAN
C     IX=SEED
C     A0,A1,A2,A3=FILTER COEFFICIENTS
C     MU=SCALING FACTOR WITHIN THE LMS ALGORITHM
C     W=WHITE GAUSSIAN NOISE GENERATED BY SUBROUTINE
C     Y=FILTER OUTPUT
C     D=DESIRED OUTPUT
C     E=ERROR(D - Y)
C
C     *******INITIAL VALUES*******
      A=1.
      MU=.01
      A0(0,0)=0.
      A1(0,0)=0.
      A2(0,0)=0.
      A3(0,0)=0.
      S=1.
      AM=0.
      IX=65539
C
C     *****OPEN FILE AND FILL INPUT MATRIX WITH WHITE
            GAUSSIAN NOISE********
      OPEN (UNIT=80,FILE='SYSID2 DATA',STATUS='NEW')
      DO 10 M=-1,31
         DO 20 N=-1,31
           IF((M.LT.0).OR.(N.LT.0))THEN
              W(M,N)=0.
           ELSE
              CALL GAUSS(IX,S,AM,V)
              W(M,N)=V*A
           ENDIF
20       CONTINUE
10    CONTINUE
C
C     ******COMPUTE THE UPDATED FILTER COEFFICIENTS
C          USING THE TWO DIMENSIONAL LMS ALGORITHM*****
      DO 40 K=0,31
         DO 50 J=0,31
           D(K,J)=.4*W(K,J)+.6*W(K-1,J)-.3*W(K,J-1)+.2*W(K-1,J-1)
           Y(K,J)=A0(K,J)*W(K,J)+A1(K,J)*W(K-1,J)+A2(K,J)*W(K,J-1)+
```

```
     *          A3(K,J)*W(K-1,J-1)
               E(K,J)=D(K,J)-Y(K,J)
               A0(K,J+1)=A0(K,J)+MU*E(K,J)*W(K,J)
               A1(K,J+1)=A1(K,J)+MU*E(K,J)*W(K-1,J)
               A2(K,J+1)=A2(K,J)+MU*E(K,J)*W(K,J-1)
               A3(K,J+1)=A3(K,J)+MU*E(K,J)*W(K-1,J-1)
50          CONTINUE
            A0(K+1,0)=A0(K,J)
            A1(K+1,0)=A1(K,J)
            A2(K+1,0)=A2(K,J)
            A3(K+1,0)=A3(K,J)
40       CONTINUE
C
C        ******OUTPUT RESULTS*******
         L=0
         DO 60 K=0,31
            DO 70 J=0,31
               L=L+1
               PRINT 15, L,E(K,J),A0(K,J),A1(K,J),A2(K,J),A3(K,J)
               WRITE (80,15) L,E(K,J),A0(K,J),A1(K,J),A2(K,J),A3(K,J)
15             FORMAT (' ',1X,I4,2X,F8.5,2X,F8.5,2X,F8.5,2X,F8.5,
     *              2X,F8.5)
70          CONTINUE
60       CONTINUE
         STOP
         END
C
         SUBROUTINE GAUSS(IX,S,AM,V)
         A=0.0
         DO 50 I=1,12
         CALL RANDU(IX,IY,Y)
         IX=IY
      50 A=A+Y
         V=(A-6.0)*S+AM
         RETURN
         END
C
C
         SUBROUTINE RANDU(IX,IY,YFL)
         IY=IX*65539
         IF(IY)5,6,6
       5 IY=IY+2147483647+1
       6 YFL=IY
         YFL=YFL*.4656613E-9
         RETURN
         END
```

APPENDIX B. LMS ALGORITHM IMPLEMENTED IN A NOISE CANCELER

```
C     THIS IS A VAX/VMS FORTRAN PROGRAM THAT IMPLEMENTS A TWO
C     DIMENSIONAL 2X2 ADAPTIVE LMS FILTER WITHIN A NOISE
C     CANCELER
C
      INTEGER M,N,K,J
      BYTE B(0:127),BYTEE(0:127),BYTEU(0:127),
      INTEGER*4 INTE(0:127,0:127),INTU(0:127,0:127),
     *          IE,IU
      REAL*4 MU,A,FMINE,FMAXE,FMINU,FMAXU
      REAL*4 AA(0:3),E(0:127,0:127),U(0:127,0:127)
      REAL*4 W(-1:127,-1:127),Y(0:127,0:127),IM(0:127,0:127)
C
C     ******VARIABLE DEFINITIONS******
C     A=NOISE AMPLITUDE
C     S=NOISE VARIANCE
C     AM=NOISE MEAN
C     AA=FILTER COEFFICIENTS
C     IX=SEED
C     MU=SCALING FACTOR WITHIN THE LMS ALGORITHM
C     IM=INPUT IMAGE
C     W=WHITE GAUSSIAN NOISE GENERATED BY SUBROUTINE
C     U=SIGNAL PLUS NOISE(IM + W)
C     Y=FILTER OUTPUT
C     E=ERROR(U - Y)
C     FMINE,FMAXE,FMINU,FAMXU=PARAMETERS
C        TO BE USED TO CONVERT DECIMAL DATA TO BYTE DATA
C
C     *******INITIAL VALUES*********
      A=1.
      MU=35.E-9
      AA(0)=0.
      AA(1)=0.
      AA(2)=0.
      AA(3)=0.
      S=40.
      AM=0.
      IX=65539
      FMINE=1.E+10
      FMAXE=-1.E+10
      FMINU=1.E+10
      FMAXU=-1.E+10
C
C     ******OPEN AN IMAGE FILE, CONVERT THE BYTE DATA INTO INTEGERS
C     AND THEN PLACE THESE VALUES IN A MATRIX*******
      OPEN (UNIT=1, NAME ='HOUS1G.DAT', TYPE ='OLD', ACCESS =
     *      'DIRECT', RECORDSIZE=32, MAXREC=128)
      DO 100 M=0,127
         READ(1'M+1) (B(J),J=0,127)
         DO 110 N=0,127
```

```
            IF(B(N).LT.0)THEN
            IM(M,N)=B(N)+256
            ELSE
            IM(M,N)=B(N)
            ENDIF
110       CONTINUE
100     CONTINUE
C
C       *******ADD WHITE GAUSSIAN NOISE TO THE IMAGE AND SET THE
C       VALUES OUTSIDE THE IMAGE TO ZERO********
        DO 10 M=-1,127
           DO 20 N=-1,127
              IF ((M.LT.0).OR.(N.LT.0))THEN
                  W(M,N)=0.
                  IM(M,N)=0.
              ELSE
                  CALL GAUSS(IX,S,AM,V)
                  W(M,N)=V*A
              ENDIF
              U(M,N)=IM(M,N)+W(M,N)
20         CONTINUE
10      CONTINUE
C
C       ******USE THE LMS ADAPTIVE ALGORITHM TO UPDATE
C       THE FILTER COEFICIENTS**********
        DO 40 K=0,127
           DO 50 J=0,127
              Y(K,J)=AA(0)*W(K,J)+AA(1)*W(K-1,J)+
     *              AA(2)*W(K,J-1)+AA(3)*W(K-1,J-1)
              E(K,J)=U(K,J)-Y(K,J)
              AA(0)=AA(0)+MU*E(K,J)*W(K,J)
              AA(1)=AA(1)+MU*E(K,J)*W(K-1,J)
              AA(2)=AA(2)+MU*E(K,J)*W(K,J-1)
              AA(3)=AA(3)+MU*E(K,J)*W(K-1,J-1)
50         CONTINUE
40      CONTINUE
C
C       *********CHANGE SIGNAL PLUS NOISE AND ERROR
C           OUTPUT INTO BYTE DATA AND THEN WRITE
            TO A FILE*********
        OPEN (UNIT=2,NAME='ERROR.DAT',TYPE='NEW',ACCESS=
     *      'DIRECT',RECORDSIZE=32,MAXREC=128)
        OPEN (UNIT=3,NAME='SIGNOISE.DAT',TYPE='NEW',ACCESS=
     *      'DIRECT',RECORDSIZE=32,MAXREC=128)
        DO 500 I=0,127
           DO 550 J=0,127
              IF(E(I,J).LT.FMINE)THEN
                FMINE=E(I,J)
              ENDIF
              IF(E(I,J).GT.FMAXE)THEN
                FMAXE=E(I,J)
              ENDIF
550        CONTINUE
500     CONTINUE
        IF (FMINE.LT.0) THEN
          FMAXE=FMAXE-FMINE
```

```
      ENDIF
      DO 600 I=0,127
         DO 650 J=0,127
            IF(FMINE.LT.0)THEN
              E(I,J)=E(I,J)-FMINE
              E(I,J)=E(I,J)*255./FMAXE
            ELSE
              E(I,J)=(E(I,J)=FMINE)*255./FMAXE
            ENDIF
            IE=NINT(E(I,J))
            IF(IE.GT.127) THEN
               IE=IE-256
            ENDIF
            BYTEE(J)-IE
650      CONTINUE
         WRITE (2'I+1) (BYTEE(N),N=0,127)
600   CONTINUE
      DO 700 I=0,127
         DO 750 J=0,127
            IF(U(I,J).LT.FMINU)THEN
              FMINU=U(I,J)
            ENDIF
            IF(U(I,J).GT.FMAXU)THEN
              FMAXU=U(I,J)
            ENDIF
750      CONTINUE
700   CONTINUE
      IF (FMINU.LT.0) THEN
        FMAXU=FMAXU-FMINU
      ENDIF
      DO 400 I=0,127
         DO 450 J=0,127
            IF(FMINU.LT.0)THEN
              U(I,J)=U(I,J)-FMINU
              U(I,J)=U(I,J)*255./FMAXU
            ELSE
              U(I,J)=(U(I,J)=FMINU)*255./FMAXU
            ENDIF
            IU=NINT(U(I,J))
            IF(IU.GT.127) THEN
               IU=IU-256
            ENDIF
            BYTEU(J)-IU
450      CONTINUE
         WRITE (3'I+1) (BYTEU(N),N=0,127)
400   CONTINUE
      CLOSE (UNIT=2)
      CLOSE (UNIT=3)
      STOP
      END
C
      SUBROUTINE GAUSS(IX,S,AM,V)
      AMP=0.0
      DO 50 I=1,12
         RANDOM=RAN(IX)
```

```
          AMP=AMP+RANDOM
50        CONTINUE
          V=(AMP-6.0)*S+AM
          RETURN
          END
```

APPENDIX C. LMS ALGORITHM IMPLEMENTED IN AN ADAPTIVE LINE ENHANCER

```
C     THIS IS A VAX/VMS FORTRAN PROGRAM THAT IMPLEMENTS A TWO
C     DIMENSIONAL 2X2 ADAPTIVE LMS FILTER WITHIN AN ADAPTIVE
C     LINE ENHANCER
C
      INTEGER M,N,K,J
      BYTE B(0:127),BYTEY(0:127),BYTEU(0:127)
      INTEGER*4 INTY(0:127,0:127),INTU(0:127,0:127),IY,IU
      REAL*4 MU,A,FMINY,FMAXY,FMINU,FMAXU
      REAL*4 AA(0:3),E(0:127,0:127),U(0:127,0:127)
      REAL*4 W(-1:127,-1:127),Y(0:127,0:127),IM(0:127,0:127)
C
C     ******VARIABLE DEFINITIONS******
C     A=NOISE AMPLITUDE
C     S=NOISE VARIANCE
C     AM=NOISE MEAN
C     AA=FILTER COEFFICIENTS
C     IX=SEED
C     MU=SCALING FACTOR WITHIN THE LMS ALGORITHM
C     IM=INPUT IMAGE
C     W=WHITE GAUSSIAN NOISE GENERATED BY SUBROUTINE
C     U=SIGNAL PLUS NOISE(IM + W)
C     Y=FILTER OUTPUT
C     WW=DELAYED VERSION OF SIGNAL PLUS NOISE(U)
C     E=ERROR(U - Y)
C     FMINY,FMAXY,FMINU,FAMXU=PARAMETERS
C        TO BE USED TO CONVERT DECIMAL DATA TO BYTE DATA
C
C     *******INITIAL VALUES**********
      A=1.
      MU=35.E-9
      AA(0)=0.
      AA(1)=0.
      AA(2)=0.
      AA(3)=0.
      S=40.
      AM=0.
      IX=65539
      FMINU=1.E+10
      FMAXU=-1.E+10
      FMINY=1.E+10
      FMAXY=-1.E+10
C
C     ******OPEN AN IMAGE FILE, CONVERT THE BYTE DATA INTO INTEGERS
C     AND THEN PLACE THESE VALUES IN A MATRIX*******
      OPEN (UNIT=1, NAME ='HOUS1G.DAT', TYPE ='OLD', ACCESS =
     *       'DIRECT', RECORDSIZE=32, MAXREC=128)
      DO 100 M=0,127
         READ(1'M+1) (B(J),J=0,127)
         DO 110 N=0,127
```

```
            IF(B(N).LT.0)THEN
            IM(M,N)=B(N)+256
            ELSE
            IM(M,N)=B(N)
            ENDIF
110       CONTINUE
100     CONTINUE
C
C       *******ADD WHITE GAUSSIAN NOISE TO THE IMAGE AND SET THE
C       VALUES OUTSIDE THE IMAGE TO ZERO*******
        DO 10 M=-1,127
           DO 20 N=-1,127
              IF ((M.LT.0).OR.(N.LT.0))THEN
                  W(M,N)=0.
                  IM(M,N)=0.
              ELSE
                  CALL GAUSS(IX,S,AM,V)
                  W(M,N)=V*A
              ENDIF
              U(M,N)=IM(M,N)+W(M,N)
              WW(M,N)=U(M-1,N)
20         CONTINUE
10      CONTINUE
C
C       ******USE THE LMS ADAPTIVE ALGORITHM TO UPDATE
C       THE FILTER COEFICIENTS**********
        DO 40 K=0,127
           DO 50 J=0,127
              Y(K,J)=AA(0)*WW(K,J)+AA(1)*WW(K-1,J)+
     *               AA(2)*WW(K,J-1)+AA(3)*WW(K-1,J-1)
              E(K,J)=U(K,J)-Y(K,J)
              AA(0)=AA(0)+MU*E(K,J)*WW(K,J)
              AA(1)=AA(1)+MU*E(K,J)*WW(K-1,J)
              AA(2)=AA(2)+MU*E(K,J)*WW(K,J-1)
              AA(3)=AA(3)+MU*E(K,J)*WW(K-1,J-1)
50         CONTINUE
40      CONTINUE
C
C       ********CHANGE SIGNAL PLUS NOISE AND FILTER OUTPUT
C            INTO BYTE DATA AND THEN WRITE TO A FILE******
        OPEN (UNIT=3,NAME='SIGNOISE.DAT',TYPE='NEW',ACCESS=
     *      'DIRECT',RECORDSIZE=32,MAXREC=128)
        OPEN (UNIT=4,NAME='FILTERED.DAT',TYPE='NEW',ACCESS=
     *      'DIRECT',RECORDSIZE=32,MAXREC=128)
        DO 700 I=0,127
           DO 750 J=0,127
              IF(U(I,J).LT.FMINU)THEN
                FMINU=U(I,J)
              ENDIF
              IF(U(I,J).GT.FMAXU)THEN
                FMAXU=U(I,J)
              ENDIF
750        CONTINUE
700     CONTINUE
        IF (FMINU.LT.0) THEN
          FMAXU=FMAXU-FMINU
```

```
      ENDIF
      DO 400 I=0,127
         DO 450 J=0,127
            IF(FMINU.LT.0)THEN
              U(I,J)=U(I,J)-FMINU
              U(I,J)=U(I,J)*255./FMAXU
            ELSE
              U(I,J)=(U(I,J)=FMINU)*255./FMAXU
            ENDIF
            IU=NINT(U(I,J))
            IF(IU.GT.127) THEN
               IU=IU-256
            ENDIF
            BYTEU(J)-IU
450      CONTINUE
         WRITE (3'I+1) (BYTEU(N),N=0,127)
400   CONTINUE
      DO 800 I=0,127
         DO 850 J=0,127
            IF(Y(I,J).LT.FMINY)THEN
              FMINY=Y(I,J)
            ENDIF
            IF(Y(I,J).GT.FMAXY)THEN
              FMAXY=Y(I,J)
            ENDIF
850      CONTINUE
800   CONTINUE
      IF (FMINY.LT.0) THEN
        FMAXY=FMAXY-FMINY
      ENDIF
      DO 900 I=0,127
         DO 950 J=0,127
            IF(FMINY.LT.0)THEN
              Y(I,J)=Y(I,J)-FMINY
              Y(I,J)=Y(I,J)*255./FMAXY
            ELSE
              Y(I,J)=(Y(I,J)=FMINY)*255./FMAXY
            ENDIF
            IY=NINT(Y(I,J))
            IF(IY.GT.127) THEN
               IY=IY-256
            ENDIF
            BYTEY(J)-IY
950      CONTINUE
         WRITE (4'I+1) (BYTEY(N),N=0,127)
900   CONTINUE
      CLOSE (UNIT=3)
      CLOSE (UNIT=4)
      STOP
      END
C
      SUBROUTINE GAUSS(IX,S,AM,V)
      AMP=0.0
      DO 50 I=1,12
         RANDOM=RAN(IX)
```

```
          AMP=AMP+RANDOM
50        CONTINUE
          V=(AMP-6.0)*S+AM
          RETURN
          END
```

APPENDIX D. RLS SYSTEM IDENTIFICATION

```
C     SYSTEM IDENTIFICATION OF AN ADAPTIVE RLS 2X2 FILTER VERSUS
C         2X2 RLS FILTER VERSUS A A 2X2 FILTER WITH
C         KNOWN COEFFICIENTS
C
      DIMENSION W(-1:32,-1:32),P(0:3,0:3),AA(0:3),X(0:3)
C
C     ******VARIABLE DEFINITIONS******
C     A=NOISE AMPLITUDE
C     S=NOISE VARIANCE
C     AM=NOISE MEAN
C     IX=SEED
C     AA=FILTER COEFFICIENTS
C     W=WHITE GAUSSIAN NOISE GENERATED BY SUBROUTINE
C     D=DESIRED OUTPUT
C     Y=FILTER OUTPUT
C     E=ERROR(D - Y)
C     MXM = ARRAY SIZE
C     NXN = FILTER SIZE
C     Q=WEIGHTING FACTOR
C     P=INVERSE OF COVARIANCE MATRIX
C
C     ******* INPUT INITIAL VALUES *************
      M=32
      N=2
      A=1.
      AA(0)=0.
      AA(1)=0.
      AA(2)=0.
      AA(3)=0.
      Q=1.
      S=1
      AM=0
      IX=65539
C
C     ****** BUILD INITIAL 'P' MATRIX **********
      DO 1 K=0,(N**2-1)
         DO 5 J=0,(N**2-1)
            IF(K.EQ.J)THEN
               P(K,J)=100.
            ELSE
               P(K,J)=0.0
            ENDIF
5        CONTINUE
1     CONTINUE
      OPEN (UNIT=63,FILE='RLSID2D2 DATA',STATUS='OLD')
C
C     ******* CALCULATE INPUT MATRIX (WHITE GAUSSIAN NOISE) ******
      DO 10 K=-1,M-1
         DO 20 J=-1,M-1
           IF((K.LT.0).OR.(J.LT.0))THEN
              W(K,J)=0.
```

```
            ELSE
               CALL GAUSS(IX,S,AM,V)
               W(K,J)=V*A
            ENDIF
20       CONTINUE
10    CONTINUE
C
C     ********* CALCULATE ERROR BETWEEN ADAPTIVE FILTER OUTPUT
C         AND KNOWN FILTER OUTPUT.  USE THIS ERROR TO UPDATE
C         FILTER COEFFICIENTS AND THE 'P' MATRIX  *************
      DO 40 K=0,M-1
         DO 50 J=0,M-1
           L=K*M+J
           D=.4*W(K,J)+.6*W(K-1,J)-.3*W(K,J-1)+.2*W(K-1,J-1)
           Y=AA(0)*W(K,J)+AA(1)*W(K-1,J)+AA(2)*W(K,J-1)+
     *          AA(3)*W(K-1,J-1)
           E=D-Y
           X(0)=W(K,J)
           X(1)=W(K-1,J)
           X(2)=W(K,J-1)
           X(3)=W(K-1,J-1)
           CALL RLS(P,X,AA,N,E,Q)
         PRINT 15, L,E,AA(0),AA(1),AA(2),AA(3)
         WRITE (63,15) L,E,AA(0),AA(1),AA(2),AA(3)
50       CONTINUE
40    CONTINUE
15    FORMAT (' ',2X,I3,2X,F9.5,2X,F9.5,2X,F9.5,2X,F9.5,2X,F9.5)
      STOP
      END
C
C
C     **********  TO COMPUTE THE GAIN MATRIX **********
      SUBROUTINE RLS(P,X,AA,N,E,Q)
      DIMENSION P(0:3,0:3),X(0:3),AA(0:3),TEMY(0:3,0:3),
     *      GAMA(0:3),TEMC(0:3)
      DO 30 L=0,3
         TEM=0.
         DO 20 K=0,3
20       TEM=TEM+X(K)*P(K,L)
30    GAMA(L)=TEM
      GAM=Q
      DO 40 K=0,3
40    GAM=GAM+GAMA(K)*X(K)
      DO 60 L=0,3
         TEM=0.
         DO 50 K=0,3
50       TEM=TEM+P(L,K)*X(K)
60    TEMC(L)=TEM
      DO 70 L=0,3
      DO 70 K=0,3
70    TEMY(L,K)=TEMC(L)*GAMA(K)
      DO 80 K=0,3
      DO 80 L=0,3
80    P(K,L)=(P(K,L)-(TEMY(K,L)/GAM))/Q
      DO 100 K=0,3
         TEM=0.
```

```
          DO 90 L=0,3
90        TEM=TEM+P(K,L)*X(L)
100     TEMC(K)=TEM
        DO 110 K=0,3
110     AA(K)=AA(K)+TEMC(K)*E
        Q=.99*Q+.01
        RETURN
        END
```

APPENDIX E. RLS ALGORITHM IMPLEMENTED IN A NOISE CANCELER

```
C     THIS IS A VAX/VMS FORTRAN PROGRAM THAT IMPLEMENTS A TWO
C     DIMENSIONAL 2X2 ADAPTIVE RLS FILTER WITHIN A NOISE
C     CANCELER
C
      INTEGER M,N,K,J
      BYTE B(0:127),BYTEE(0:127),BYTEU(0:127)
      INTEGER*4 INTE(0:127,0:127),INTU(0:127,0:127),IE,IU
      REAL*4 A,FMINE,FMAXE,FMINU,FMAXU
      REAL*4 AA(0:3),E(0:127,0:127),U(0:127,0:127)
      REAL*4 P(0:3,0:3),X(0:3),IM(0:127,0:127)
      REAL*4 W(-1:127,-1:127),Y(0:127,0:127)
C
C     ******VARIABLE DEFINITIONS******
C     A=NOISE AMPLITUDE
C     S=NOISE VARIANCE
C     AM=NOISE MEAN
C     AA=FILTER COEFFICIENTS
C     IX=SEED
C     IM=INPUT IMAGE
C     W=WHITE GAUSSIAN NOISE GENERATED BY SUBROUTINE
C     U=SIGNAL PLUS NOISE(IM + W)
C     Y=FILTER OUTPUT
C     E=ERROR(U - Y)
C     Q=WEIGHTING FACTOR
C     P=INVERSE OF COVARIANCE MATRIX
C     FMINE,FMAXE,FMINU,FAMXU=PARAMETERS
C        TO BE USED TO CONVERT DECIMAL DATA TO BYTE DATA
C     MXM=ARRAY SIZE
C     NXN=FILTER SIZE
C
C     *******INITIAL VALUES*********
      M=128
      N=2
      Q=1.
      A=1.
      AA(0)=0.
      AA(1)=0.
      AA(2)=0.
      AA(3)=0.
      S=40.
      AM=0.
      IX=65539
      FMINE=1.E+10
      FMAXE=-1.E+10
      FMINU=1.E+10
      FMAXU=-1.E+10
```

```
C
C     ******OPEN AN IMAGE FILE, CONVERT THE BYTE DATA INTO INTEGERS
C     AND THEN PLACE THESE VALUES IN A MATRIX*******
      OPEN (UNIT=1, NAME ='HOUS1G.DAT', TYPE ='OLD', ACCESS =
     *       'DIRECT', RECORDSIZE=32, MAXREC=128)
      DO 100 K=0,127
         READ(1'K+1) (B(L),L=0,127)
         DO 110 J=0,127
            IF(B(J).LT.0)THEN
            IM(K,J)=B(J)+256
            ELSE
            IM(K,J)=B(J)
            ENDIF
110      CONTINUE
100   CONTINUE
      CLOSE (UNIT=1)
C
C     ****** BUILD INITIAL 'P' MATRIX *********
      DO 1 K=0,(N**2-1)
         DO 5 J=0,(N**2-1)
            IF(K.EQ.J)THEN
               P(K,J)=100.
            ELSE
               P(K,J)=0.0
            ENDIF
5        CONTINUE
1     CONTINUE
C
C     *******ADD WHITE GAUSSIAN NOISE TO THE IMAGE AND SET THE
C     VALUES OUTSIDE THE IMAGE TO ZERO*******
      DO 10 K=-1,M-1
         DO 20 J=-1,M-1
            IF ((K.LT.0).OR.(J.LT.0))THEN
                W(K,J)=0.
                IM(K,J)=0.
            ELSE
                CALL GAUSS(IX,S,AM,V)
                W(K,J)=V*A
            ENDIF
            U(K,J)=IM(K,J)+W(K,J)
20       CONTINUE
10    CONTINUE
C
C     ********** CALCULATE ERROR BETWEEN ADAPTIVE FILTER OUTPUT
C         AND KNOWN FILTER OUTPUT.  USE THIS ERROR TO UPDATE
C         FILTER COEFFICIENTS AND THE 'P' MATRIX  **************
      DO 40 K=0,M-1
         DO 50 J=0,M-1
           L=K*M+J
           Y=AA(0)*W(K,J)+AA(1)*W(K-1,J)+AA(2)*W(K,J-1)+
     *         AA(3)*W(K-1,J-1)
           E(K,J)=U(K,J)-Y(K,J)
           EE=E(K,J)
           X(0)=W(K,J)
           X(1)=W(K-1,J)
           X(2)=W(K,J-1)
```

```
            X(3)=W(K-1,J-1)
            CALL RLS(P,X,AA,N,EE,Q)
50        CONTINUE
40      CONTINUE
C
C       ********CHANGE SIGNAL PLUS NOISE AND ERROR OUTPUT
C            INTO BYTE DATA AND WRITE TO A FILE*****
        OPEN (UNIT=2,NAME='ERROR.DAT',TYPE='NEW',ACCESS=
       *     'DIRECT',RECORDSIZE=32,MAXREC=128)
        OPEN (UNIT=3,NAME='SIGNOISE.DAT',TYPE='NEW',ACCESS=
       *     'DIRECT',RECORDSIZE=32,MAXREC=128)
        DO 500 I=0,127
           DO 550 J=0,127
              IF(E(I,J).LT.FMINE)THEN
                FMINE=E(I,J)
              ENDIF
              IF(E(I,J).GT.FMAXE)THEN
                FMAXE=E(I,J)
              ENDIF
550        CONTINUE
500     CONTINUE
        IF (FMINE.LT.0) THEN
          FMAXE=FMAXE-FMINE
        ENDIF
        DO 600 I=0,127
           DO 650 J=0,127
              IF(FMINE.LT.0)THEN
                E(I,J)=E(I,J)-FMINE
                E(I,J)=E(I,J)*255./FMAXE
              ELSE
                E(I,J)=(E(I,J)=FMINE)*255./FMAXE
              ENDIF
              IE=NINT(E(I,J))
              IF(IE.GT.127) THEN
                 IE=IE-256
              ENDIF
              BYTEE(J)-IE
650        CONTINUE
           WRITE (2'I+1) (BYTEE(N),N=0,127)
600     CONTINUE
        DO 700 I=0,127
           DO 750 J=0,127
              IF(U(I,J).LT.FMINU)THEN
                FMINU=U(I,J)
              ENDIF
              IF(U(I,J).GT.FMAXU)THEN
                FMAXU=U(I,J)
              ENDIF
750        CONTINUE
700     CONTINUE
        IF (FMINU.LT.0) THEN
          FMAXU=FMAXU-FMINU
        ENDIF
        DO 400 I=0,127
           DO 450 J=0,127
              IF(FMINU.LT.0)THEN
```

```
              U(I,J)=U(I,J)-FMINU
              U(I,J)=U(I,J)*255./FMAXU
            ELSE
              U(I,J)=(U(I,J)=FMINU)*255./FMAXU
            ENDIF
            IU=NINT(U(I,J))
            IF(IU.GT.127) THEN
               IU=IU-256
            ENDIF
            BYTEU(J)-IU
450       CONTINUE
          WRITE (3'I+1) (BYTEU(N),N=0,127)
400     CONTINUE
        CLOSE (UNIT=2)
        CLOSE (UNIT=3)
        STOP
        END
C
C       ********** TO COMPUTE THE GAIN MATRIX **********
        SUBROUTINE RLS(P,X,AA,N,E,Q)
        DIMENSION P(0:3,0:3),X(0:3),AA(0:3),TEMY(0:3,0:3),
       *      GAMA(0:3),TEMC(0:3)
        DO 30 L=0,3
           TEM=0.
           DO 20 K=0,3
20         TEM=TEM+X(K)*P(K,L)
30      GAMA(L)=TEM
        GAM=Q
        DO 40 K=0,3
40      GAM=GAM+GAMA(K)*X(K)
        DO 60 L=0,3
           TEM=0.
           DO 50 K=0,3
50         TEM=TEM+P(L,K)*X(K)
60      TEMC(L)=TEM
        DO 70 L=0,3
        DO 70 K=0,3
70      TEMY(L,K)=TEMC(L)*GAMA(K)
        DO 80 K=0,3
        DO 80 L=0,3
80      P(K,L)=(P(K,L)-(TEMY(K,L)/GAM))/Q
        DO 100 K=0,3
           TEM=0.
           DO 90 L=0,3
90         TEM=TEM+P(K,L)*X(L)
100     TEMC(K)=TEM
        DO 110 K=0,3
110     AA(K)=AA(K)+TEMC(K)*E
        Q=.99*Q+.01
        RETURN
        END
```

APPENDIX F. RLS ALGORITHM IMPLEMENTED IN AN ADAPTIVE LINE ENHANCER

```
C     THIS IS A VAX/VMS FORTRAN PROGRAM THAT IMPLEMENTS A TWO          APP00010
C     DIMENSIONAL 2X2 ADAPTIVE RLS FILTER WITHIN AN ADAPTIVE           APP00020
C     LINE ENHANCER                                                    APP00030
C                                                                      APP00040
      INTEGER M,N,K,J                                                  APP00050
      BYTE B(0:127),BYTEU(0:127),BYTEY(0:127)                          APP00060
      INTEGER*4 INTY(0:127,0:127),INTU(0:127,0:127),IU,IY              APP00070
      REAL*4 A,FMINU,FMAXU,FMINY,FMAXY                                 APP00080
      REAL*4 AA(0:3),E(0:127,0:127),U(0:127,0:127)                     APP00090
      REAL*4 P(0:3,0:3),X(0:3),IM(0:127,0:127)                         APP00100
      REAL*4 W(-1:127,-1:127),Y(0:127,0:127)                           APP00110
C                                                                      APP00120
C     ******VARIABLE DEFINITIONS******                                 APP00130
C     A=NOISE AMPLITUDE                                                APP00140
C     S=NOISE VARIANCE                                                 APP00150
C     AM=NOISE MEAN                                                    APP00160
C     AA=FILTER COEFFICIENTS                                           APP00170
C     IX=SEED                                                          APP00180
C     IM=INPUT IMAGE                                                   APP00190
C     W=WHITE GAUSSIAN NOISE GENERATED BY SUBROUTINE                   APP00200
C     U=SIGNAL PLUS NOISE(IM + W)                                      APP00210
C     Y=FILTER OUTPUT                                                  APP00220
C     E=ERROR(U - Y)                                                   APP00230
C     WW=DELAYED VERSION OF SIGNAL PLUS NOISE(U)                       APP00240
C     Q=WEIGHTING FACTOR                                               APP00250
C     P=INVARSE OF COVARIANCE MATRIX                                   APP00260
C     FMINU,FAMXU,FMINY,FMAXY=PARAMETERS                               APP00270
C        TO BE USED TO CONVERT DECIMAL DATA TO BYTE DATA               APP00280
C     MXM=ARRAY SIZE                                                   APP00290
C     NXN=FILTER SIZE                                                  APP00300
C                                                                      APP00310
C     ********INITIAL VALUES**********                                APP00320
      M=128                                                            APP00330
      N=2                                                              APP00340
      Q=1.                                                             APP00350
      A=1.                                                             APP00360
      AA(0)=0.                                                         APP00370
      AA(1)=0.                                                         APP00380
      AA(2)=0.                                                         APP00390
      AA(3)=0.                                                         APP00400
      S=40.                                                            APP00410
      AM=0.                                                            APP00420
      IX=65539                                                         APP00430
      FMINU=1.E+10                                                     APP00440
      FMAXU=-1.E+10                                                    APP00450
      FMINY=1.E+10                                                     APP00460
      FMAXY=-1.E+10                                                    APP00470
C                                                                      APP00480
```

```
C     ******OPEN AN IMAGE FILE, CONVERT THE BYTE DATA INTO INTEGERS      APP00490
C     AND THEN PLACE THESE VALUES IN A MATRIX*******                    APP00500
      OPEN (UNIT=1, NAME ='HOUS1G.DAT', TYPE ='OLD', ACCESS =           APP00510
     *       'DIRECT', RECORDSIZE=32, MAXREC=128)                       APP00520
      DO 100 K=0,127                                                    APP00530
         READ(1'K+1) (B(L),L=0,127)                                     APP00540
         DO 110 J=0,127                                                 APP00550
            IF(B(J).LT.0)THEN                                           APP00560
            IM(K,J)=B(J)+256                                            APP00570
            ELSE                                                        APP00580
            IM(K,J)=B(J)                                                APP00590
            ENDIF                                                       APP00600
110      CONTINUE                                                       APP00610
100   CONTINUE                                                          APP00620
      CLOSE (UNIT=1)                                                    APP00630
C                                                                       APP00640
C     ****** BUILD INITIAL 'P' MATRIX **********                       APP00650
      DO 1 K=0,(N**2-1)                                                 APP00660
         DO 5 J=0,(N**2-1)                                              APP00670
            IF(K.EQ.J)THEN                                              APP00680
               P(K,J)=100.                                              APP00690
            ELSE                                                        APP00700
               P(K,J)=0.0                                               APP00710
            ENDIF                                                       APP00720
5        CONTINUE                                                       APP00730
1     CONTINUE                                                          APP00740
C                                                                       APP00750
C     *******ADD WHITE GAUSSIAN NOISE TO THE IMAGE AND SET THE          APP00760
C     VALUES OUTSIDE THE IMAGE TO ZERO*******                           APP00770
      DO 10 K=-2,M-1                                                    APP00780
         DO 20 J=-2,M-1                                                 APP00790
            IF ((K.LT.0).OR.(J.LT.0))THEN                               APP00800
                W(K,J)=0.                                               APP00810
                IM(K,J)=0.                                              APP00820
            ELSE                                                        APP00830
                CALL GAUSS(IX,S,AM,V)                                   APP00840
                W(K,J)=V*A                                              APP00850
            ENDIF                                                       APP00860
            U(K,J)=IM(K,J)+W(K,J)                                       APP00870
20       CONTINUE                                                       APP00880
10    CONTINUE                                                          APP00890
C                                                                       APP00900
C     *****COMPUTE THE DELAYED VERSION OF THE SIGNAL                    APP00910
C         PLUS NOISE MATRIX(U)******                                    APP00920
      DO 45 K=-1,127                                                    APP00930
         DO 46 J=-1,127                                                 APP00940
            IF((M.LT.-1).OR.(N.LT.-1) THEN                              APP00950
              WW(K,J)=0.                                                APP00960
            ELSE                                                        APP00970
              WW(K,J)=U(K-1,J)                                          APP00980
            ENDIF                                                       APP00990
46       CONTINUE                                                       APP01000
45    CONTINUE                                                          APP01010
C                                                                       APP01020
C     ********** CALCULATE ERROR BETWEEN ADAPTIVE FILTER OUTPUT         APP01030
```

```
C          AND KNOWN FILTER OUTPUT.  USE THIS ERROR TO UPDATE          APP01040
C          FILTER COEFFICIENTS AND THE 'P' MATRIX  **************          APP01050
      DO 40 K=0,M-1                                                      APP01060
         DO 50 J=0,M-1                                                   APP01070
           Y=AA(0)*WW(K,J)+AA(1)*WW(K-1,J)+AA(2)*WW(K,J-1)+              APP01080
     *        AA(3)*WW(K-1,J-1)                                          APP01090
           E(K,J)=U(K,J)-Y(K,J)                                          APP01100
           EE=E(K,J)                                                     APP01110
           X(0)=WW(K,J)                                                  APP01120
           X(1)=WW(K-1,J)                                                APP01130
           X(2)=WW(K,J-1)                                                APP01140
           X(3)=WW(K-1,J-1)                                              APP01150
           CALL RLS(P,X,AA,N,EE,Q)                                       APP01160
50       CONTINUE                                                        APP01170
40    CONTINUE                                                           APP01180
C                                                                        APP01190
C     ********CHANGE SIGNAL PLUS NOISE AND FILTER OUTPUT                 APP01200
C          INTO BYTE DATA AND WRITE TO A FILE*****                       APP01210
      OPEN (UNIT=3,NAME='SIGNOISE.DAT',TYPE='NEW',ACCESS=                APP01220
     *     'DIRECT',RECORDSIZE=32,MAXREC=128)                            APP01230
      OPEN (UNIT=4,NAME='FILTERED.DAT',TYPE='NEW',ACCESS=                APP01240
     *     'DIRECT',RECORDSIZE=32,MAXREC=128)                            APP01250
      DO 700 I=0,127                                                     APP01260
         DO 750 J=0,127                                                  APP01270
            IF(U(I,J).LT.FMINU)THEN                                      APP01280
              FMINU=U(I,J)                                               APP01290
            ENDIF                                                        APP01300
            IF(U(I,J).GT.FMAXU)THEN                                      APP01310
              FMAXU=U(I,J)                                               APP01320
            ENDIF                                                        APP01330
750      CONTINUE                                                        APP01340
700   CONTINUE                                                           APP01350
      IF (FMINU.LT.0) THEN                                               APP01360
        FMAXU=FMAXU-FMINU                                                APP01370
      ENDIF                                                              APP01380
      DO 400 I=0,127                                                     APP01390
         DO 450 J=0,127                                                  APP01400
            IF(FMINU.LT.0)THEN                                           APP01410
              U(I,J)=U(I,J)-FMINU                                        APP01420
              U(I,J)=U(I,J)*255./FMAXU                                   APP01430
            ELSE                                                         APP01440
              U(I,J)=(U(I,J)=FMINU)*255./FMAXU                           APP01450
            ENDIF                                                        APP01460
            IU=NINT(U(I,J))                                              APP01470
            IF(IU.GT.127) THEN                                           APP01480
               IU=IU-256                                                 APP01490
            ENDIF                                                        APP01500
            BYTEU(J)-IU                                                  APP01510
450      CONTINUE                                                        APP01520
         WRITE (3'I+1) (BYTEU(N),N=0,127)                                APP01530
400   CONTINUE                                                           APP01540
      DO 800 I=0,127                                                     APP01550
         DO 850 J=0,127                                                  APP01560
            IF(Y(I,J).LT.FMINY)THEN                                      APP01570
              FMINY=Y(I,J)                                               APP01580
            ENDIF                                                        APP01590
```

```
            IF(Y(I,J).GT.FMAXY)THEN                                    APP01600
              FMAXY=Y(I,J)                                             APP01610
            ENDIF                                                      APP01620
850       CONTINUE                                                     APP01630
800     CONTINUE                                                       APP01640
        IF (FMINY.LT.0) THEN                                           APP01650
          FMAXY=FMAXY-FMINY                                            APP01660
        ENDIF                                                          APP01670
        DO 900 I=0,127                                                 APP01680
           DO 950 J=0,127                                              APP01690
              IF(FMINY.LT.0)THEN                                       APP01700
                Y(I,J)=Y(I,J)-FMINY                                    APP01710
                Y(I,J)=Y(I,J)*255./FMAXY                               APP01720
              ELSE                                                     APP01730
                Y(I,J)=(Y(I,J)=FMINY)*255./FMAXY                       APP01740
              ENDIF                                                    APP01750
              IY=NINT(Y(I,J))                                          APP01760
              IF(IY.GT.127) THEN                                       APP01770
                 IY=IY-256                                             APP01780
              ENDIF                                                    APP01790
              BYTEY(J)-IY                                              APP01800
950       CONTINUE                                                     APP01810
          WRITE (4'I+1) (BYTEY(N),N=0,127)                             APP01820
900     CONTINUE                                                       APP01830
        CLOSE (UNIT=3)                                                 APP01840
        CLOSE (UNIT=4)                                                 APP01850
        STOP                                                           APP01860
        END                                                            APP01870
C                                                                      APP01880
C       **********  TO COMPUTE THE GAIN MATRIX **********             APP01890
        SUBROUTINE RLS(P,X,AA,N,E,Q)                                   APP01900
        DIMENSION P(0:3,0:3),X(0:3),AA(0:3),TEMY(0:3,0:3),             APP0191C
       *      GAMA(0:3),TEMC(0:3)                                      APP01920
        DO 30 L=0,3                                                    APP01930
           TEM=0.                                                      APP01940
           DO 20 K=0,3                                                 APP01950
20         TEM=TEM+X(K)*P(K,L)                                         APP01960
30      GAMA(L)=TEM                                                    APP01970
        GAM=Q                                                          APP01980
        DO 40 K=0,3                                                    APP01990
40      GAM=GAM+GAMA(K)*X(K)                                           APP02000
        DO 60 L=0,3                                                    APP02010
           TEM=0.                                                      APP02020
           DO 50 K=0,3                                                 APP02030
50         TEM=TEM+P(L,K)*X(K)                                         APP02040
60      TEMC(L)=TEM                                                    APP02050
        DO 70 L=0,3                                                    APP02060
        DO 70 K=0,3                                                    APP02070
70      TEMY(L,K)=TEMC(L)*GAMA(K)                                      APP02080
        DO 80 K=0,3                                                    APP02090
        DO 80 L=0,3                                                    APP02100
80      P(K,L)=(P(K,L)-(TEMY(K,L)/GAM))/Q                              APP02110
        DO 100 K=0,3                                                   APP02120
           TEM=0.                                                      APP02130
```

```
          DO 90 L=0,3                                          APP02140
90        TEM=TEM+P(K,L)*X(L)                                  APP02150
100     TEMC(K)=TEM                                            APP02160
        DO 110 K=0,3                                           APP02170
110     AA(K)=AA(K)+TEMC(K)*E                                  APP02180
        Q=.99*Q+.01                                            APP02190
        RETURN                                                 APP02200
        END                                                    APP02210
```

LIST OF REFERENCES

1. Andrews, H. and Hunt, B., *Digital Image Restoration*, Prentice-Hall, Inc., Reading, Massachusetts, 1987.

2. Gonzalez, R. and Wintz, P., *Digital Image Processing*, Addison-Wesley Publishing Co., Englewood Cliffs, New Jersey, 1977.

3. Rosenfeld, A. and Kak, A., *Digital Picture Processing*, Academic Press, New York, 1976.

4. Murphy, P. K., *Survey of Image Restoration Techniques*, The Johns Hopkins University, Applied Physics Laboratory, Laurel, Maryland, July 1988.

5. Treichler, J., Johnson, C., and Larimore, M., *Theory and Design of Adpaptive Filters*, John Wiley and Sons Inc., New York, 1984.

6. Orfanidis, S. J., *Optimum Signal Processing, an Introduction*, Macmillian, 1985.

7. Giordano, A. and Hsu, F., *Least Square Estimation with Applications to Digital Signal Processing*, John Wiley and Sons Inc., New York, 1985.

8. Haykin, S., *Introduction To Adaptive Filters*, Macmillian Publishing Company, New York, 1984.

9. Hadhoud, M. M. and Thomas D. W., "The Two-Dimensional Adaptive LMS Algorithm," *IEEE Trans. Circuits Syst.*, Vol. CAS-35, no. 5, pp. 485-494, May 1988.

10. Ekstrom, M. P., "Realizable Wiener Filtering in Two Dimensions," *IEEE Trans. Acoust., Speech, Signal Processing*, Vol. ASSP-30, no. 30, pp. 31-39, February 1982.

11. Florian, S. and Feuer, A., "Performance Analysis of the LMS Algorithm with a Tapped Delay Line (Two Dimensional Case)," *IEEE Trans. Acoust., Speech, Signal Processing*, Vol. ASSP-34, no. 6, pp. 1542-1549, December 1986.

12. Bitmead, R. R. and Anderson B. D. O., "Performance of Adaptive Estimation Algorithms in Dependent Random Environments," Environments," *IEEE Trans. Automatic Control*, Vol. AC-25, pp. 788-794, August 1980.

13. Widrow, B. and Stearns, S. D., *Adaptive Signal Processing*, Prentice-Hall, Inc., Englewood Cliffs, New Jersey, 1985.

14. Wellstead, P. E. and Caldas Pintos J. R., "Self-tuning Filters and Predictors for Two Dimensional Systems," *International Journal of Control*, Vol. 42, no. 2, pp. 457-478, 1985.

INITIAL DISTRIBUTION LIST

		No. Copies
1.	Defense Technical Information Center Cameron Station Alexandria, VA 22304-6145	2
2.	Library, Code 0142 Naval Postgraduate School Monterey, CA 93943-5002	2
3.	Chairman Electrical and Computer Engineering Department, Code 62 Naval Postgraduate School Monterey, CA 93940-5000	1
4.	Dr. Murali Tummala Electrical and Computer Engineering Department, Code 62 Naval Postgraduate School Monterey, CA 93940-5000	2
5.	Dr. Charles Therrien Electrical and Computer Engineering Department, Code 62 Naval Postgraduate School Monterey, CA 93940-5000	1
6.	Professor Will Gersch Department of Information and Computer Sciences University of Hawaii Honolulu, HI 96822	1
7.	Patrol Squadron Forty-Seven Attn: Lcdr. A. R. Topp FPO San Francisco San Francisco, CA 96601-5920	1
8.	Naval Underwater Systems Center Attn: Lt. R. E. Reinke New London, CT 06320	1
9.	Patrol Squadron Nine Attn: Lcdr. Steven L. Wilstrup FPO San Francisco San Francisco, CA 96601-5905	1
10.	Ocean Systems Center Commander Naval Ocean Systems Center San Diego, CA 92152-5000	1